Clamor

Speech Recognition

MOTOROLA

What you never thought possi

Clamor™ Noise-Robust Speech Recognition

Motorola, Lexicus Division offers *Clamor* - noise-robust, speaker-dependent, small vocabulary speech recognition software. Clamor is designed to be particularly insensitive to background noise and requires little memory. This makes it suitable for embedded applications such as speaker phones, organizers and PDAs, home stereos, set-top boxes, toys and video games, and automotive and industrial applications. Clamor is available on several platforms: the ARM family of microprocessors from Advanced RISC Machines; the OS-9 Real-Time Operating System from Microware Systems Corporation; Motorola's DSP 56166 and DSP 56800; and Memos™, Motorola's new operating environment for messaging products. A Windows® demo and DLL are available for prototyping embedded applications or for inclusion in Windows software products.

FEATURES

High Accuracy in Any Environment
- 99.6+% accuracy in a normal office
- 96+% accuracy in extreme noise

Extremely Noise Robust
- Works well in cars with the windows open, in airport lobbies, or with music blasting.

Reliability
- Press-to-talk guarantees reliable recognition

Flexible Vocabulary & Number of Users
- Up to 40 phrases active
- Unlimited number of switchable dictionaries
- Only two repetitions needed to train a phrase

Works Across Languages
- Speaker-dependent, language independent
- American and British English, French, German, Farsi, Japanese, Cantonese, and Mandarin tested

A free evaluation demo can be downloaded from the Lexicus web site at http://www.mot.com/lexicus

EMBEDDED SYSTEMS

Ported to Multiple Operating Systems
- API available for ARM, OS-9, Memos, and Windows platforms. C code can be easily ported to other operating systems.

Small Memory Footprint (2K ROM 3.5K RAM)
- 10 word vocabulary occupies only 5.5K memory for code, data, & buffering

WINDOWS DLL

Fast Response Time
- 0.15 seconds on a Pentium 133 chip; on-going optimization will improve this response time
- Clamor API is available with documentation
- 22K DLL, plus .2K per template second

Platform Requirements
- 486 or Pentium, Windows 95, and Windows NT
- Most standard omni-directional microphones
- SoundBlaster®-compatible sound card

LEXICUS RECOGNIZERS

Handwriting Recognition
- Lexicus also offers handwriting recognition in English and Chinese for embedded systems and Windows platforms

Motorola's DSP563xx Instruction Set Simulator

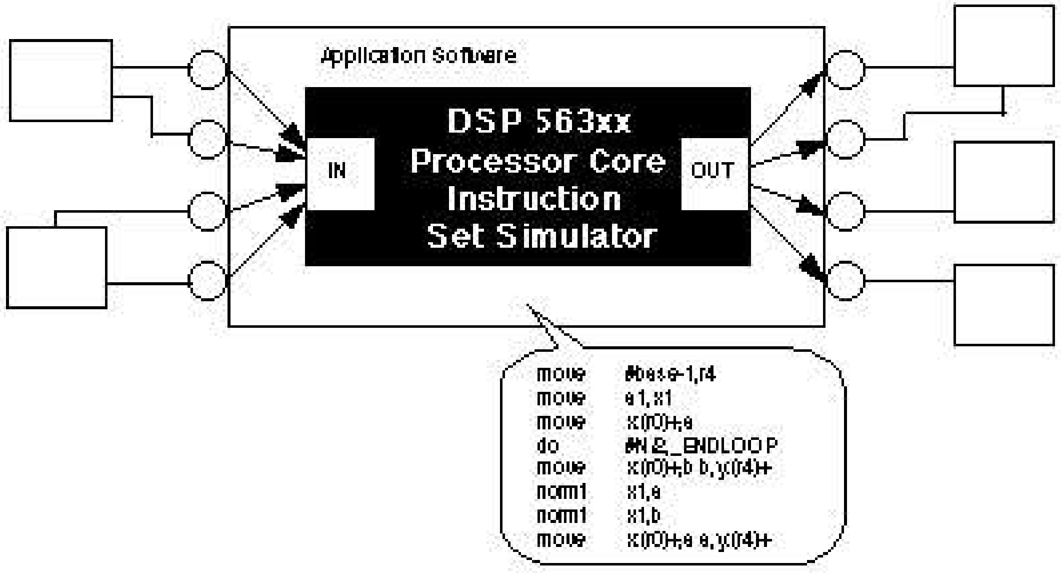
Alta's Signal Processing WorkSystem (SPW)

Bridging the GAP between DSP hardware and software design.

- ▲ Easy integration of user-specified assembly code into a system level model.
- ▲ Fast functional verification of assembly code within the context of the system.
  - SPW generated system test vectors
  - Easy analysis of results ~ visualization with SigCalc/ISL
- ▲ Software testing within the system environment through the integrated debug environment.
  - Single step
  - Examine/Set registers
  - Read/Modify memory

The integration of Motorola's DSP563xx Instruction set simulator into Alta's SPW enables easy assembly code development, debug and verification within the context of the complete system design.

SPW™ is a block-oriented design, simulation, and implementation environment for electronic systems. Common application areas for SPW include wireless and wired communications, multimedia, and networking. SPW is ideal for algorithm development, filter design, C code generation, hardware/software architecture co-design and hardware synthesis. SPW is built around Alta's CONVERGENCE simulation architecture, which combines datapath and control simulation within a single environment. This makes SPW the first commercially available solution capable of capturing and simulating today's most advanced electronic products and systems.



```
move    #base-1,r4
move    a1,x1
move    x(r0)+,a
do      #N2_ENDLOOP
move    x(r0)+,b b,y(r4)+
norm1   x1,a
norm1   x1,b
move    x(r0)+,a a,y(r4)+
```

cādence

Alta Business Unit

Headquarters
555 N. Mathilda Avenue
Sunnyvale, CA 94086
408-733-1595 Fax: 408-523-4601
http://www.atlagroup.com

ANALOGICAL SYSTEMS

Algorithm Selection Guide

Speech and audio compression algorithms for the DSP56000/DSP56300 families:

Algorithm	Bit Rate (kbps)	Description	Availability
G.729 CS-ACELP	8	ITU-T Conjugate-Structure Algebraic-Code-Excited Linear Prediction (CS-ACELP)	Q4'97
G.723.1 ACELP/MP-MLQ	5.3, 6.	ITU-T Algebraic-Code-Excited Linear Prediction (ACELP) Multi-Pulse Maximum Likelihood Quantization (MP-MLQ)	Now
G.728 LD-CELP	1	ITU-T Low-Delay, Code-Excited Linear Prediction (LD-CELP) speech and audio compression	Now
G.726 ADPCM	16, 24, 32, 40	ITU-T Adaptive Differential Pulse Code Modulation (ADPCM) audio compression	Now
G.711 A-law/µ-law	48, 56, 64	A-law/µ-law commanding routines	Now
USFS 1016 CELP	4.8	Code-Excited Linear Prediction (CELP)	Now
Multirate CELP	4.8 to 8.0	CELP with compile-time bit rate selection	Now

Telecommunications algorithms for the DSP56000/DSP56300 families:

Algorithm	Description	Availability
Echo Canceler	Line echo canceler. Fully G.165 compliant.	Now
DTMF Detector	Dual tone multifrequency (DTMF) signaling detector. Fully Q.24 compliant. ZERO hits on the MItel talkoff test tape. Single or multichannel.	Now
MF(R1) Detector	Signaling system R1 multifrequency (MF) tone detector. Fully Q.323 compliant. Single or multichannel.	Now
MF(R2) Detector	Signaling system R2 multifrequency (MF) tone detector. Fully Q.455 compliant. Single or multichannel.	Now
Call Progress Tone Detector	Single or multichannel call progress monitor. Supports dial tone, ringback, PBX double ringback, busy, and reorder. Well-suited for use on international lines.	Now
Caller ID Detector	Bellcore Caller ID detector	Now

Source code for all algorithms is available under a one-time, royalty-free license arrangement. Source code packages include:

▲ Complete, professional documentation

▲ Well-commented source code

▲ Technical support

▲ Future software upgrades

▲ Custom integration services available

ANALOGICAL SYSTEMS

Real-Time DSP Solutions

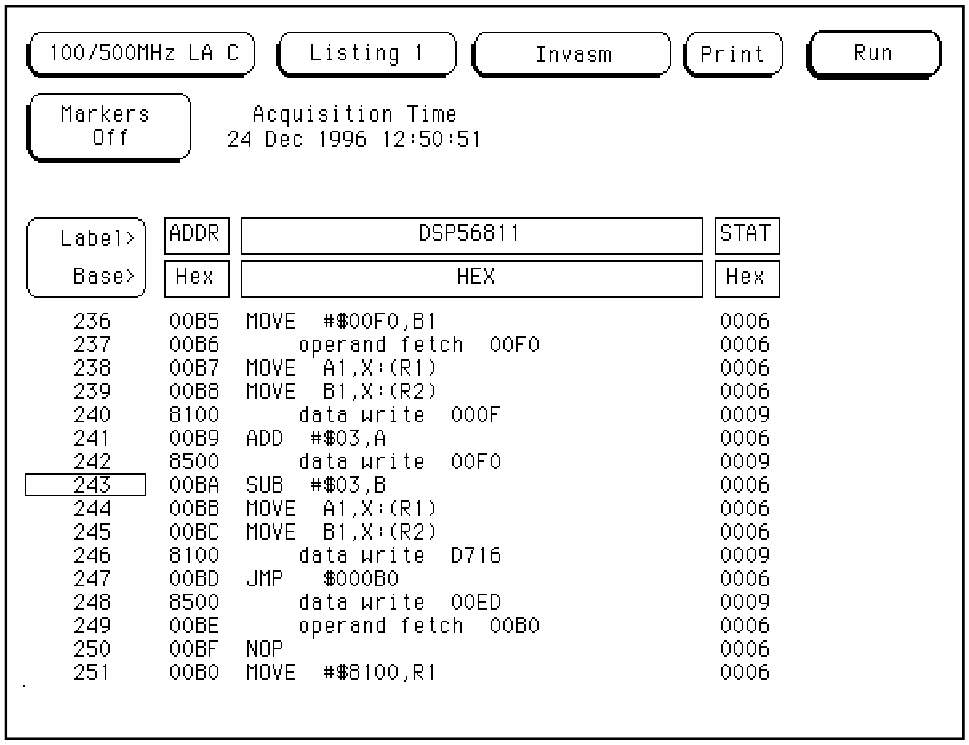
Contact Information
Analogical Systems
299 California Avenue
Palo Alto, CA 94306
Tel: (415) 323-3232
Fax: (415) 323-4222
E-mail: info@analogical.com

CORELIS

HEWLETT PACKARD
Premier Solution Partner

PI-56L811
PREPROCESSOR INTERFACE
FOR USE WITH
HP16500/1660/1670 LOGIC ANALYZERS

- **COMPATIBLE WITH MOTOROLA DSP56L811 DSP CHIP**
- **COMPLETE DSP56L811MNEMONIC DISASSEMBLY**
- **DISPLAY OF CYCLE STATUS INFORMATION INCLUDING IDENTIFICATION OF INSTRUCTIONS FETCH AND OPERAND READ/WRITE CODES**
- **QUICK AND EASY CONNECTION OF LOGIC ANALYZER PODS TO A DSP56L811 TARGET SYSTEM**
- **LOW CAPACITANCE PROBING**
- **SET-UP AND DATA STORAGE ON BUILT-IN LOGIC ANALYZER DISK DRIVE**
- **TRACE DATA HARD COPY VIA RS-232 SERIAL PORT**
- **MULTI-LAYER, LOW NOISE PCB CONSTRUCTION WITH GROUND AND POWER PLANES**



```
Acquisition Time
24 Dec 1996 12:50:51

Label>   ADDR   DSP56811                          STAT
Base>    Hex    HEX                               Hex
 236     00B5   MOVE  #$00F0,B1                   0006
 237     00B6       operand fetch   00F0          0006
 238     00B7   MOVE   A1,X:(R1)                  0006
 239     00B8   MOVE   B1,X:(R2)                  0006
 240     8100       data write   000F             0009
 241     00B9   ADD   #$03,A                      0006
 242     8500       data write   00F0             0009
 243     00BA   SUB   #$03,B                      0006
 244     00BB   MOVE   A1,X:(R1)                  0006
 245     00BC   MOVE   B1,X:(R2)                  0006
 246     8100       data write   D716             0009
 247     00BD   JMP    $000B0                     0006
 248     8500       data write   00ED             0009
 249     00BE       operand fetch   00B0          0006
 250     00BF   NOP                               0006
 251     00B0   MOVE  #$8100,R1                   0006
```

The PI-56L811 Preprocessor Interface provides a complete interface between any DSP56L811 target system and the HP16500, HP1660, or HP1670 family of logic analyzers. The PI-56L811 configuration software on a flexible disk sets up the format specification menu of the logic analyzer for compatibility with the DSP56L811 microprocessor. It also loads the inverse assembler (disassembler) for obtaining displays of the DSP56L811 data in DSP56L811 assembly language mnemonics. The PI-56L811 Preprocessor Interface is a non-intrusive development tool and provides a powerful environment for debugging of both hardware and software real-time applications.

General Overview

The PI-56L811 Preprocessor Interface is a specialized module that provides a convenient interface between the HP16500/1660/1670 family of logic analyzers and an DSP56L811 target system.

The PI-56L811 Preprocessor is attached directly to the DSP56L811 processor, thus eliminating any need to remove the processor from the target board. The logic analyzer pods, with HP01650-63203 termination adapters, plug directly onto the mating connectors on the PI-56L811 and provide tracing and monitoring of the DSP56L811 signals. The signals are grouped in a logical order so that the HP logic analyzer configured with the disassembler software can display bus activity in mnemonic form. In addition to the mnemonic disassembly, the logic analyzer displays all the bus activity with the relevant status information. The preprocessor supports the 100 pin TQFP package.

Hewlett Packard logic analyzers use built-in 3.5 inch floppy-disk drive(s) to store set-ups, trace configuration and data for later use. The PI-56L811 Preprocessor Interface includes a floppy disk that contains the DSP56L811 inverse assembler and the configuration set-up.

Hewlett-Packard logic analyzers are part of an integrated family of design and development tools. Many different models are available and include networking capability, oscilloscope add-ons, the ability to display high-level source code, and many other features. Please see your local HP sales rep for additional information.

Specifications

● **Logic Analyzer Required**
Hewlett-Packard HP16500, HP1660, and HP1670 family of analyzers

● **Maximum Acquisition Speed**
The maximum aquisition speed is only limited by the speed of the logic analyzer.

● **Signal Line Loading**
20pF @ 100K Ohms

● **Number of Probes Required**
Four, sixteen channel probes are required for complete state disassembly.

● **Microprocessor Package Supported**
Supports the 100 pin TQFP package

● **Included**
- PI-56L811 Preprocessor Interface module
- Disassembler and configuration software diskette
- Operating manual

CORELIS 12607 Hidden Creek Way
Cerritos, California 90703
Tel: (562) 926-6727, Fax: (562) 404-6196
sales@corelis.com
http: //www.corelis.com

DSPworks

DIGITAL SIGNAL PROCESSING APPLICATION

Momentum Data Systems
17330 Brookhurst St. Ste. 140
Fountain Valley, CA 92708
Phone: (714) 557-6884, Fax (714) 557-6969, email: dsp@mds.com, web: http://www.mds.com

General Features:

- DATA ACQUISITION
- SIGNAL GENERATION
- SIGNAL ANALYSIS
- WAVEFORM SYNTHESIS
- DSP FUNCTIONS
- SPECTRAL DISPLAY
- WATERFALL DISPLAY
- FILE IMPORT/EXPORT
- SCRIPT FEATURES
- REAL-TIME OSCILLOSCOPE
- HARDWARE ACCELERATORS

DSPworks is a multi-platform general-purpose signal processing application which is designed to provide powerful signal processing features.

The following platforms are currently supported:

- PC under Windows
- UNIX-based workstations

The system is completely menu-driven and easy to use - in fact, several of our customers have stated that they have never needed to open the accompanying reference guide. Because of DSPworks' intuitive characteristics, the learning curve is minimized. It was designed with both the novice and experienced user in mind - there are no commands to memorize and learn before you're up and running.

DSPworks provides the DSP professional a complete library of built-in DSP algorithms for a variety of data and signal analysis operations. As with all our products, we are committed to providing the very best DSP software available at an affordable price.

DSP Board Support:

It is our objective to offer the most extensive board support possible, and currently support about thirty different DSP boards. We continue to expand the range of boards supported, so please call if you would like to discuss your particular requirements with us.

Digital Signal Processing Capabilities

Generators:

Waveform synthesis takes place in the Generators section of the program which contains a large variety of functions for generating discrete data sequences as follows:

- Sinusoidal
- Swept Sine
- Square
- Triangular
- Exponential
- Windowing Functions
- Unit Step
- Sinc
- Ramp
- Unit Sample
- Noise

Noise can be added to waveform based on probability density functions. Signal length is limited only by disk capacity of the system disk. Signals are real valued only. Signals can be generated as 32-bit floating point values or 16-bit fractional fixed point values.

DSP Functions:

- **Signal Filtering**
  Apply filter designed by QEDesign to a time domain sequence
- **Convolution**
- **Autocorrelation**
- **Crosscorrelation**
- **Decimation**
- **Interpolation**
- **Fast Fourier Transform**
- **Inverse FFT**
- **Multirate Filter**
- **Sample Rate Conversion**
- **Average FFT**
- **Inverse Wiener Filter**

Display Capabilities:

- **Waveform display**

  Display any stored waveforms. Stored waveforms can be either generated or acquired signals.
- **Real-time Oscilloscope Display**
- **Spectral Display**

  One-dimensional Magnitude or Power display. Normally used for real-time display. Can be used to display stored waveform. Phase can also be displayed.
- **Two/Three Dimensional Spectral Display**

  Two/Three dimensional magnitude or power display. Classic spectogram or sonogram normally used for real-time display. Can be used to display stored waveform. Three dimensional is classic waterfall display.

Operations on Generated or Acquired Signals:

- Arithmetic
- Reciprocal
- Square
- Square Root
- Shift
- Flip
- Join
- Trigonometric
- Exponential
- Extract
- Smooth
- Sample & Hold
- Difference
- Quantize Fixed Point
- Statistics

Miscellaneous Features

- **File Import/Export**
- **Play Script**

  Scripts may be recorded and stored for convenience.
- **Record to Disk and Playback**

  Record in DSPworks format on system disk from external source through a DSP card. Acquired signals will be 16-bit fractional fixed point.
- **ASCII/Binary Format Conversion**
- **Integer/ Float Data Type Conversion**
- **Demultiplex/Multiplex Operations**

  For separating and combining multi-channel signals.
- **Hardware Accelerator**

  Optional hardware accelerator functions using available DSP card(s) for FFT; Inverse FFT; Filtering; Convolution; Autocorrelation; Crosscorrelation

QEDesign

Digital Filter Design and Analysis Application

Momentum Data Systems
17330 Brookhurst St. Ste 140
Fountain Valley, CA 92708
Tel: (714) 557-6884
Fax: (714) 557-6969

E-mail: dsp@mds.com

Web: http://www.mds.com

QEDesign is an easy-to-use, multi-platform advanced digital filter design package. It is completely menu-driven and user intuitive, affording almost a non-existent start-up and learning curve. QEDesign performs complex mathematical computations for filter design, provides superior graphical displays, and generates comprehensive design reports.

General Features

QEDesign Series features include:

- completely menu-driven system
- extensive error-checking
- extensive on-line help features
- use of 64-bit floating point for all calculations
- use of 128-bit floating point for critical design areas
- coefficient quantization variable from 8 to 32 bits
- coefficient scaling
- recycling of input for comparative analysis
- tiled and stacked graphic displays
- specification file for retention of previously designed filters
- transfer function analysis

Hardware Platforms

The following platforms are currently supported:

- PC (DOS and Windows)
- Macintosh
- Engineering workstations (Xwindows/Motif)

Three versions are available:

- QEDesign 500
- QEDesign 1000
- QEDesign 2000

QEDesign 500 is a basic filter design system available for either DOS-or Windows. QEDesign 1000 is a more sophisticated version with extended features. QEDesign 1000 is available as a standard DOS application as well as a Windows 3.X application. A Macintosh version is also available.QEDesign 2000 is our premier filter design system for engineering workstations and offers additional features over QEDesign 1000.

Code Generators

Momentum Data Systems offers a complete line of Code Generators to complement QEDesign's filter design capabilities. These code generators are designed to work seamlessly with QEDesign and provides the ability to produce assembly code quickly and easily.

The code generation module is accessible through a pull-down menu and reads coefficient files generated by QEDesign. It then creates highly optimized assembly language programs for both IIR and FIR filters.

General Features

- Modular programs for easy modification of input/output programs
- Complete programs including interrupt processing and handling of analog input/output

Code Generators Available:

- Motorola DSP56001
- Motorola DSP56002
- Motorola DSP56004
- Motorola DSP56005
- Motorola DSP56156
- Motorola DSP56166
- Motorola DSP96002
- Motorola M68HC16

Extensive range of boards supported

DEMOS
AVAILABLE
UPON
REQUEST

Infinite Impulse Response Design

Infinite Impulse Response (IIR) digital filter design means that the sample output is a function of previous outputs as well as the current and previous input samples. The transfer function for such a filter has both poles and zeros. The poles must be within the unit circle in the Z-domain for a stable filter.

IIR filters can be designed in the analog domain (S plane) and then mapped to the digital domain (Z plane) or they can be designed directly in the Z plane. QEDesign provides five types of analog filter prototypes and three methods of transforming an S plane design to the digital domain. QEDesign also provides an allpass filter with arbitrary group delay capability. This filter is designed directly in the Z plane.

Each of the design calculations requires large numbers of numerical calculations.

In order to provide accurate coefficients for any filter order, QEDesign performs all design calculations in at least 64-bit floating point. Some very critical calculations in QEDesign 2000 for the Sun Workstations are performed in 128-bit precision.

After calculating the coefficients with great accuracy, the coefficients must be quantized to a specific word length for implementation in a digital signal processor.

QEDesign provides complete quantization analysis. Quantizing the coefficients perturbs the location of the poles and zeros, so QEDesign shows the effects of this perturbation in the graphical displays of the filter characteristics. QEDesign also provides detailed analysis of the effects of finite arithmetic operations and can compute the output noise power, the least significant bit without error and the dynamic range of the filter.

- Lowpass, Highpass, Bandpass, Bandstop Filters, Arbitrary Group Delay
- Filter orders:

– Lowpass	80
– Highpass	80
– Bandpass	160
– Bandstop	160
– Arbitrary Group Delay	160

- Analog Prototype Filters:
  - Butterworth
  - Tschebyscheff
  - Inverse Tschebyscheff
  - Elliptic
  - Bessel
- Digital Transformation methods:
  - Bilinear Transformation
  - Impulse Invariant
  - Matched Z-Transform
- Optional Phase Equalization
- Graphical Output includes:
  - Magnitude
  - Log Magnitude
  - Poles and Zeroes
  - Impulse Response
  - Phase
  - Group Delay
  - Step Response
- Quantization Features
  - Quantize Coefficients (8-32 bits)
  - Coefficient Scaling to prevent overflow
  - Computation of Dynamic Range
  - Computation of Least Significant Bit in Error
  - Output Noise Power Calculation
  - Analysis of Finite Arithmetical Operations
- Coefficients can be scaled for the following realizations:
  - Cascade Form 2 for fixed point implementation
  - Transpose of Cascade Form 2 for fixed point implementation
  - Parallel Form 1 for fixed point implementation
  - Cascade and parallel forms for floating point implementation
  - Direct form (ratio of polynomials)
- Reports show design details such as all transformations from normalized lowpass filter to desired filter coefficients

Finite Impulse Response Design

Finite Impulse Response (FIR) Design means that the sample output is a function of the current and previous input samples only. Previous output samples do not in any way affect the current sample output. The transfer function for this type of filter consists of zeros only and as a result, FIR filters are always stable.

FIR filters are normally assumed to be linear phase i.e. the group delay is constant. This is true only if the filter coefficients have certain symmetries. QEDesign will create linear phase filters only, thus all FIR filters are either symmetric or antisymmetric about their center point.

There are several methods of designing FIR filters. QEDesign supports the most useful methods - window design and Parks-McClellan design.

Since all frequency functions are periodic on the unit circle of the z-domain, the magnitude and phase are periodic functions in the frequency domain. Thus it is possible to represent these functions as a Fourier series with the coefficients of the Fourier series representing the coefficients of the filter. To form a causal filter, the Fourier series is truncated and shifted.

The truncation of the Fourier series causes a phenomenon called the "Gibbs effect". This is a spike that occurs wherever there is a discontinuity in the desired magnitude of the filter. To counteract this, the filter coefficients are convolved in the frequency domain with the spectrum of a window function thus smoothing the edge transitions at any discontinuity. This convolution in the frequency domain is equivalent to multiplying the filter coefficients with the window coefficients giving the final filter coefficients.

QEDesign provides a large number of windows with both fixed and variable falloff to the first sidelobe in the magnitude response.

Parks-McClellan (Equiripple)

The Parks-McClellan design method uses an optimization algorithm called the Remez Exchange Algorithm. This type of design normally produces equiripple designs whereby the ripples in the passbands and stopbands are of equal height in any one band.

QEDesign has options for most filter types to alter this characteristic and allows Rolloff values to be specified in 3dB increments. The optimization algorithm utilizes 64-bit precision arithmetic for all calculations. This is essential in the design of long filters.

Both types of FIR design (window functions and Parks-McClellan) allow specification of either symmetric or antisymmetric filters. This, coupled with the option of specifying transition band functions, can lead to unique designs such as antisymmetric bandpass filter with root raised cosine transition functions.

- Filter Types
  - Lowpass
  - Highpass
  - Bandpass
  - Bandstop,
  - Differentiator
  - Multiband
  - Hilbert Transformer
  - Arbitrary Magnitude
  - Halfband
  - Raised Cosine
  - Root Raised Cosine Filters
- Filter Orders

– Parks-McClellan	4089
– Window Design	8192

- Available Window Functions:
  - Rectangular
  - Hanning (Hann)
  - Hamming
  - Triangular
  - Blackman
  - Exact Blackman
  - 3 Term Cosine
  - 3 Term Cosine with continuous 3rd Derivative
  - Minimum 3 Term Cosine
  - 4 Term Cosine
  - 4 Term Cosine with continuous 5th Derivative
  - Minimum 4 Term Cosine
  - Good 4 Term Blackman Harris
  - Harris Flat Top
  - Kaiser
  - Dolph-Tschebyscheff
  - Taylor
  - Gaussian
- Graphical output includes:
  - Magnitude
  - Log Magnitude
  - Impulse Response
  - Step Response
- Coefficient Quantization from 8-32 bits
- Reports show design details
- Filters can be designed for a nominal gain of 1 or maximum gain of 1
- Sin(x)/x Compensation
- Comb filter compensation
- Specification of Transition Regions on Selected Filter Types
- Choice of Symmetric/Antisymmetric FIR Filters

System Analysis

The System Analysis section of the system allows one to determine the characteristics (Magnitude, Phase, Group Delay, Impulse Response, Pole/Zero locations, and Step Response) of a given transfer function.

The transfer function can be input in the z-domain as:

- A ratio of polynomials
- Zeros & Poles
- Product of second order sections
- Sum of second order sections
- Symmetric FIR Filter
- Antisymmetric FIR Filter

A transfer function specified in the s-domain (i.e. Analog Transfer function) can be specified as:

- Ratio of Polynomials
- Zero and Poles
- Product of second order sections

Graphical Design

- A unique feature is the graphical design via adding or deleting poles and zeros graphically and moving existing poles and zeros. This design capability is sometimes needed to design filters that cannot be specified in a conventional manner. This feature also builds intuition on the result of placement of poles and zeros in the z domain.
- Placement of poles and zeros via mouse input, simultaneous display of system responses while moving poles or zeros
- Selection of either rectangular or polar coordinates and zoom-in/out capability for precise placement of poles/zeros.

Comparison of features for Momentum’s various QEDesign filter design packages:

	500 (PC)	1000 (PC/Mac)	2000 (UNIX)
Filter Types			
Lowpass	Y	Y	Y
Highpass	Y	Y	Y
Bandpass	Y	Y	Y
Bandstop	Y	Y	Y
Differentiator		Y	Y
Multiband		Y	Y
Hilbert Transformer		Y	Y
Arbitrary Magnitude (FIR)		Y	Y
Halfband		Y	Y
Raised Cosine		Y	Y
Root Raised Cosine		Y	Y
Arbitrary Group Delay		Y	Y
Filter Orders			
IIR	40	80	160
FIR with windows	1024	3072	8192
Equiripple FIR	256	1380	4089
IIR Analog Prototypes			
Butterworth	Y	Y	Y
Tschebyscheff	Y	Y	Y
Inverse Tschebyscheff	Y	Y	Y
Elliptic	Y	Y	Y
Bessel	Y	Y	Y
Digital Transformation Methods			
Bilinear	Y	Y	Y
Impulse Invariant		Y	Y
Matched Z-Transform			Y
IIR Realization Methods for bilinear and matched z-transform designs			
Cascade	Y	Y	Y
Parallel			Y
Ratio of polynomials			Y
FIR Window Functions			
Rectangular	Y	Y	Y
Triangular	Y	Y	Y
Hanning	Y	Y	Y
Hamming	Y	Y	Y
Blackman	Y	Y	Y
Kaiser	Y	Y	Y
Exact Blackman		Y	Y
3 Term Cosine		Y	Y
3 Term Cosine with continuous 3rd Derivative		Y	Y
Minimum 3 Term Cosine		Y	Y
4 Term Cosine		Y	Y
4 Term Cosine with continuous 5th Derivative		Y	Y
Minimum 4 Term Cosine		Y	Y
Good 4 Term Blackman Harris		Y	Y
Harris Flat Top		Y	Y
Dolph-Tschebyscheff		Y	Y
Taylor		Y	Y
Gaussian		Y	Y
System Analysis			
Z-Domain transfer function	Y	Y	Y
S-Domain transfer function		Y	Y
Compensation Features			
D to A conversion		Y	Y
Comb Filter Compensation		Y	Y
Phase Equalization		Y	Y
Plot Options			
Magnitude	Y	Y	Y
Log Magnitude	Y	Y	Y
Poles and Zeros	Y	Y	Y
Phase	Y	Y	Y
Group Delay	Y	Y	Y
Impulse Response	Y	Y	Y
Step Response		Y	Y
Logarithmic Frequency Scale		Y	Y
Graphical Placement of Poles and Zeros with simultaneous display of system response			Y
Quad Precision Calculations for critical designs			Y

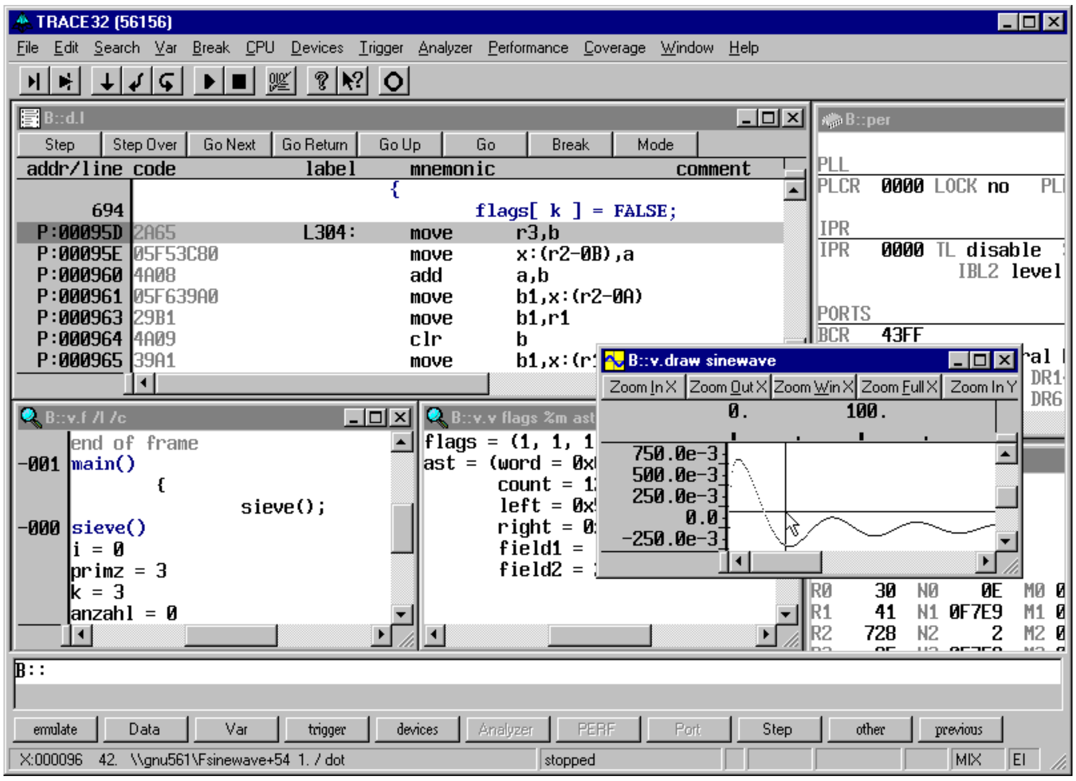
Momentum Data Systems, 17330 Brookhurst St. Ste 140, Fountain Valley, CA 92708, Ph: (714) 557-6884, Fax: (714) 557-6969
E-mail: dsp@mds.com Web: http://www.mds.com

TRACE32®

DSP56k

TRACE32-ICD

for Motorola's DSP56k


TRACE32 (56156)
File Edit Search Var Break CPU Devices Trigger Analyzer Performance Coverage Window Help
B::d.l
Step Step Over Go Next Go Return Go Up Go Break Mode
B::per
B::v.f /l /c
B::v.v flags %m ast
B::v.draw sinewave
Zoom In X Zoom Out X Zoom Win X Zoom Full X Zoom In Y
0. 100.
750.0e-3 500.0e-3 250.0e-3 0.0 -250.0e-3
emulate Data Var trigger devices Analyzer PERF Port Step other previous

```
addr/line code        label    mnemonic        comment
                                {
      694                          flags[ k ] = FALSE;
P:00095D 2A65         L304:    move     r3,b
P:00095E 05F53C80              move     x:(r2-0B),a
P:000960 4A08                  add      a,b
P:000961 05F639A0              move     b1,x:(r2-0A)
P:000963 29B1                  move     b1,r1
P:000964 4A09                  clr      b
P:000965 39A1                  move     b1,x:(r
```

```
PLL
PLCR   0000 LOCK no    PL
IPR
IPR    0000 TL disable
            IBL2 level
PORTS
BCR    43FF
```

```
      end of frame
-001 main()
         {
                   sieve();
-000 sieve()
     i = 0
     primz = 3
     k = 3
     anzahl = 0
```

```
flags = (1, 1, 1
ast = (word = 0x
        count = 1
        left = 0x
        right = 0
        field1 =
        field2 =
```

```
R0   30  N0    0E  M0 0
R1   41  N1 0F7E9  M1 0
R2  728  N2     2  M2 0
```

```
B::
```

```
X:000096  42.  \\gnu561\Fsinewave+54  1. / dot     stopped     MIX  EI
```

TRACE32 provides a complete set of development tools and covers all solutions in testing software and hardware for the embedded market. TRACE32 tools are designed to significatly reduce the time spent testing and debugging. The powerful TRACE32 window interface offers a seamless integration with the entire range of TRACE32 hardware.

TRACE32-ICD for the DSP56k family is a largely cost effective tool for debugging on high level language and assembler level. The implementation of the TRACE32 In-Circuit Debuggers is based on the ONCE interface and all features provided by the ONCE port are fully supported.

TRACE32-ICD DSP56k Features:

- Easy high-level debugging
- Easy debugging on assembler level
- Graphical variable display
- Program- and Spot-breakpoints
- Breakpoints on Variables
- Display of internal and external peripherals at a logical level
- Flash Programming
- Trigger In- and Output
- Powerful script language
- Very fast download
- Connection via TRACE32-ICE or directly to the host system

TRACE32®

http://www.lauterbachbach.com

Processor Supported:

DSP56002/4/5, DSP561xx, DSP563xx, DSP566xx, DSP568xx and DPS-part of the 68356

Third Party Compiler Supported:

Motorola, Tasking

Host Interfaces:

Stand-alone:	WINDOWS 95/NT
Ethernet:	WINDOWS 95/NT SUN/Solaris HP/UX LINUX DEC Alpha

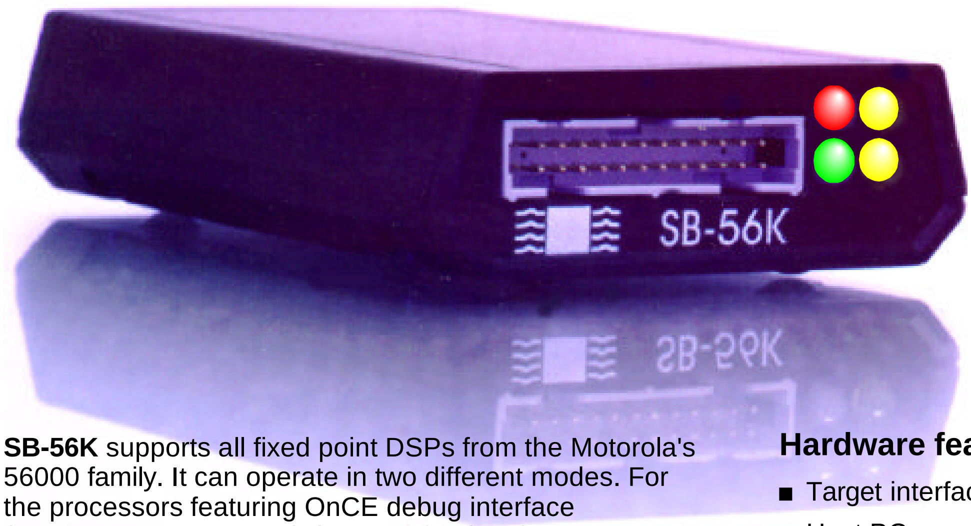
Contact:

Lauterbach Inc.	***Lauterbach Datentechnik GmbH***
5 Mount Royal Ave.	Fichtenstr. 27
Marlborough, MA 01752	D-85649 Hofolding
Tel: (508) 303 6812	Tel: ++49 8104 8943-0
Fax: (508) 303 6813	Fax: ++49 8104 8943-30
Email: info_us@lauterbach.com	Email: info@lauterbach.com

LAUTERBACH

Universal Emulator for Motorola's DSPs

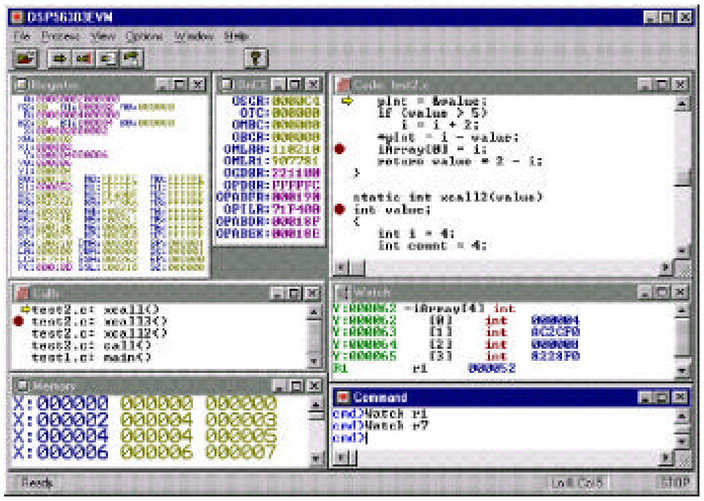
SoftBox Corp.



SB-56K supports all fixed point DSPs from the Motorola's 56000 family. It can operate in two different modes. For the processors featuring OnCE debug interface (DSP56000, DSP56100), it allows for the simultaneous acces to two devices at the same time. For the JTAG interface (DSP56300, DSP56600 and DSP56800) number of DSPs is unlimited (for the higher quantities of the target devices, buffering of the JTAG signals can be necessary). SB-56K communicates with the host computer through the high-speed RS-232 connection. Basic user interface is shown below. User application is available for Windows 95 or NT (DOS, Windows 3.1 and Win32s are not supported). Optional software allows for the operation over the TCP/IP network (LAN, WAN or Internet). Connection between **SB-56K** - controlling target DSPs and User interface(s) is provided through the windows sockets. Individual DSPs can be accessed from the multiple workstations.

Hardware features:

- Target interface: JTAG or OnCE (double)
- Host PC connection: RS-232
- Emulation clock controlled by the emulator: 8MHz - DC
- JTAG/OnCE logic implemented in the FPGA
- Controller based on 60 MHz DSP56002 with 128Kword of RAM and 128 Kbytes of Flash ROM
- Firmware loaded during system initialization (no ROMs, jumpers nor switches)
- Counter for external events (up to 100 MHz)
- Small size - dimensions: 4.5" x 2.5" x 1"
- Power requirements: 400 mA @ 5 VDC



```
OSCR:0000C1
 OTC:000000
OMBC:000000
OBCR:000000
OMLR0:110218
OMLR1:507781
OGDBR:221100
OPDBR:FFFFFC
OPABFR:000198
OPILR:71F400
OPABDR:00018F
OPABEX:00018E
```

```
    pInt = &value;
    if (value > 5)
        i = i + 2;
    *pInt = i - value;
    iArray[0] = i;
    return value * 2 - i;
}

static int xcall2(value)
int value;
{
    int i = 4;
    int count = 4;
```

```
test2.c: xcall()
test2.c: xcall3()
test2.c: xcall2()
test2.c: call()
test1.c: main()
```

```
Y:000062 -iArray[4] int
Y:000062    [0]     int   000004
Y:000063    [1]     int   8C2CF0
Y:000064    [2]     int   000000
Y:000065    [3]     int   8224F0
R1          r1      000052
```

```
X:000000 000000 000000
X:000002 000004 000003
X:000004 000004 000005
X:000006 000006 000007
```

```
cmd>Watch r1
cmd>Watch r7
cmd>
```

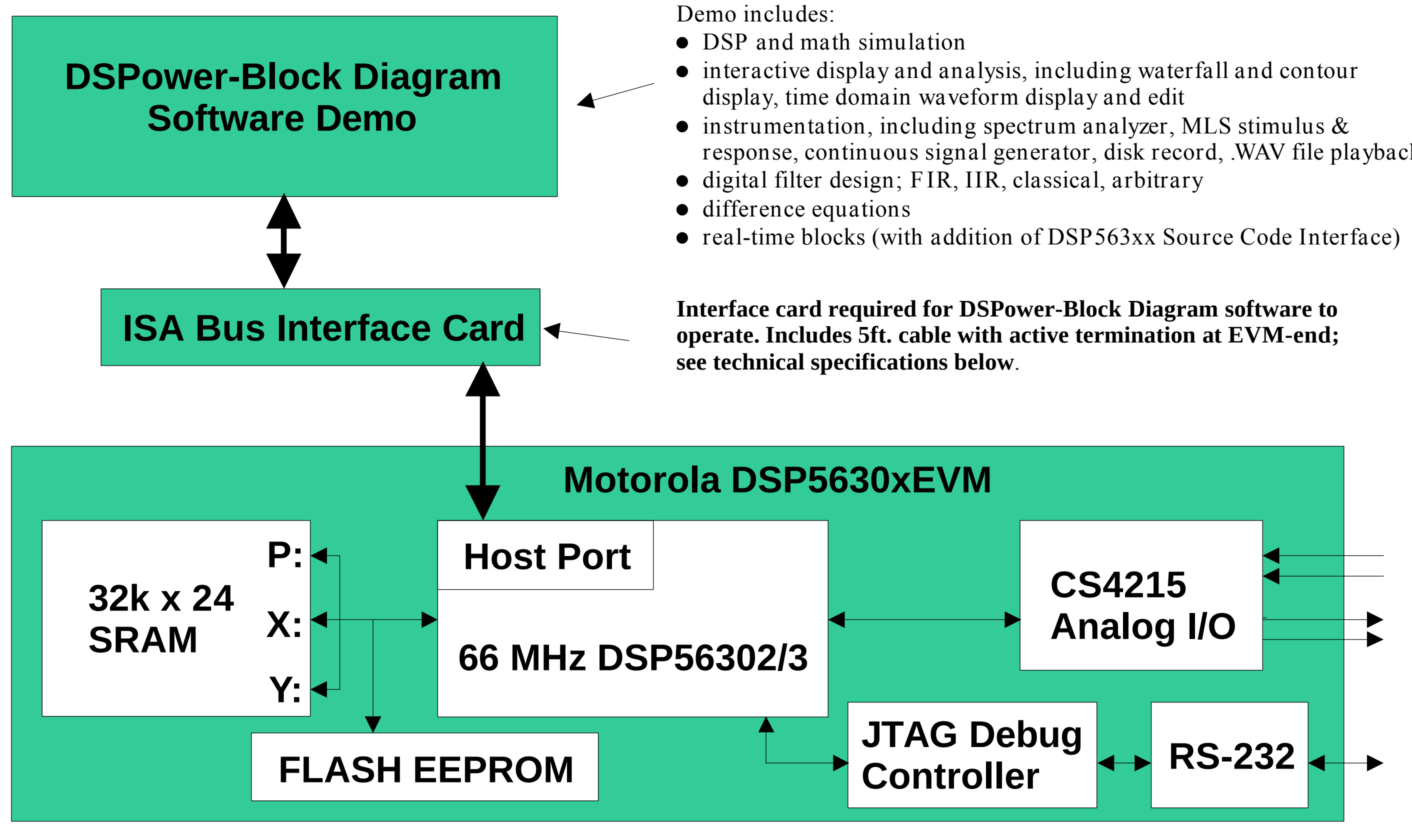
Software features:

- One application per target DSP (BoxView)
- Multiple Data, Code, Register windows
- User configurable buttons and controls
- Local communication through system pipes
- Optional client-server TCP/IP connections (BoxView + BoxServer)
- Core software available for licensing to third parties

SoftBox Corp.
1621 Scottsdale Dr., Plano, TX 75023
Tel.: (972) 519-1665, Fax: (972) 964-5417
E-mail: info@softbox.com
http://www.softbox.com

DSPower®-Block Diagram Software Demo + Motorola 5630xEVM

Data Sheet



Demo includes:

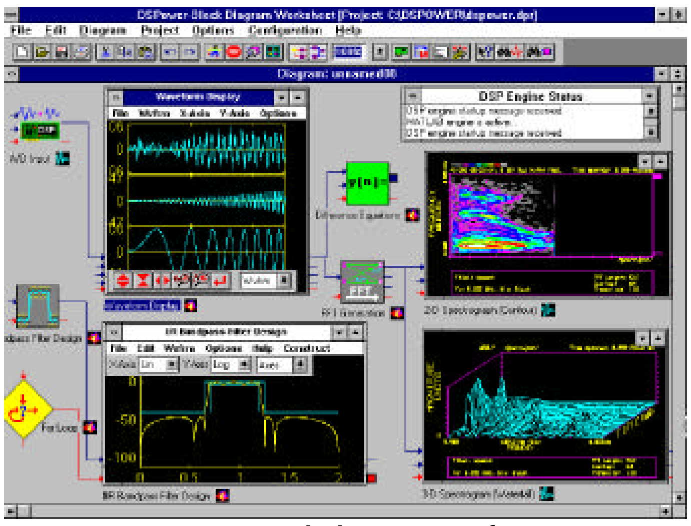
- DSP and math simulation
- interactive display and analysis, including waterfall and contour display, time domain waveform display and edit
- instrumentation, including spectrum analyzer, MLS stimulus & response, continuous signal generator, disk record, .WAV file playback
- digital filter design; FIR, IIR, classical, arbitrary
- difference equations
- real-time blocks (with addition of DSP563xx Source Code Interface)

Interface card required for DSPower-Block Diagram software to operate. Includes 5ft. cable with active termination at EVM-end; see technical specifications below.

Software Overview

Signalogic software supports Motorola 5630xEVM boards at several levels:

- DSPower-Block Diagram is a Windows program which offers a block diagram-based DSP design environment, including both block-diagram simulation and interactive display and instrument functions. Blocks are implemented as Hypersignal® macro language or MATLAB® .m files; display and instrument blocks include toolbar, popup dialog, cursor control, and other interactive objects. User-defined blocks are straightforward to add.
- The DSPower Real-Time Code Generator allows C source code generated from DSP and math blocks, combined with user-defined C code, to be compiled and downloaded to the 5630xEVM board for real-time execution.
- Hypersignal-Macro and Hypersignal-Acoustic software series, which offer a number of simulation and real-time instrument functions. Simulation functions include DSP and math functions, time domain waveform display and editing, frequency domain display (including waterfall, contour, magnitude, unwrapped phase), difference equations, digital FIR and IIR filter design, sampling rate conversion, frequency zoom, wavelet transform, minimum phase calculation, and many more. Instrument functions include spectrum analyzer, digital oscilloscope, stimulus & response measurement, continuous signal generation, real-time "snap-in" filtering, continuous disk record and generate, and more.

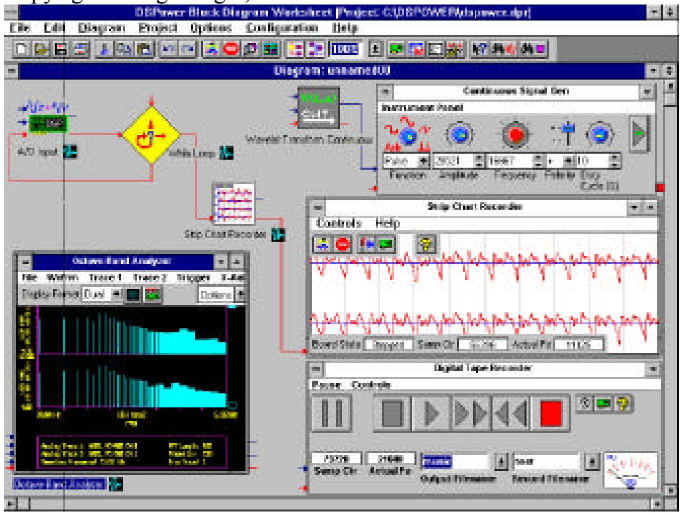

DSPower Block Diagram Worksheet [Project: C:\DSPOWER\dspower.dpr]
File Edit Diagram Project Options Configuration Help
Diagram: unnamed00
Waveform Display
DSP Engine Status
A/D Input
Difference Equations
FFT
FFT Generation
2-D Spectrogram (Contour)
IIR Bandpass Filter Design
For Loop
3-D Spectrogram (Waterfall)

DSPower-Block Diagram Software
Algorithm Simulation and Real-Time Code Generation

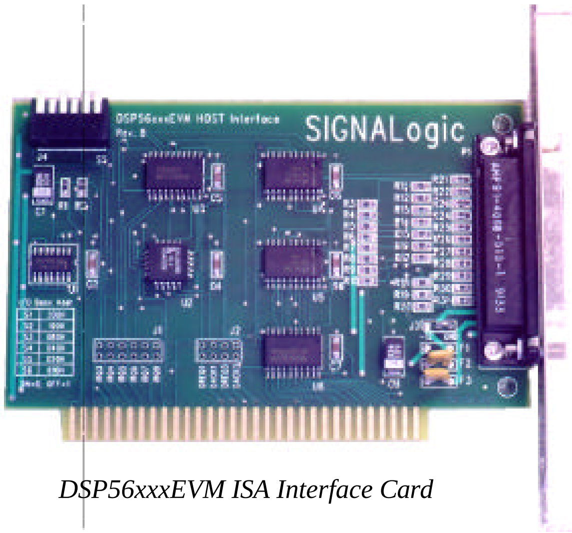
- DSPower®-HWLib software provides a C/C++, MATLAB, or Visual Basic programming interface to the 5630xEVM. It offers a library of low-level and high-level calls for user-defined program including low-level 5630xEVM functions such as reset/run/hold,

SIGNALOGIC®

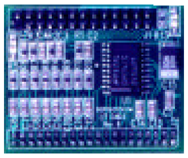
9617 Wendell @ Skillman • Dallas, TX 75243
tel: 214-343-0069 • fax: 214-343-0163
e-mail: dspinfo@signalogic.com • web: www.signalogic.com



Example Data Acquisition Functions in DSPower



DSP56xxxEVM ISA Interface Card



5630x Active Termination

register access, block memory transfer, COFF (executable) file download, etc. High-level functions include waveform file acquire/generate, continuous signal generation, and execution of arbitrary Hypersignal DSP or math functions. DSPower-HWLib includes digital oscilloscope and digital tape recorder C++ programs as examples, and also a basic debugger that allows display of DSP memory contents in graphical and text formats.

- The DSP56xxx Source Code Interface contains numerous DSP algorithms and functions in source and binary form, such as optimized FFTs, filters, matrix, tanscendental, trig, signal manipulation functions, 5630xEVM initialization and analog I/O examples, etc. These functions form the basis of many Hypersignal functions and instruments; modification of these can be used to customize Hypersignal or DSPower-HWLib operation. The DSP56xxx Source Code Interface can also be used as a basic foundation for any user-defined DSP system or product.

Developing code for the DSP56303 for use with Signalogic software packages requires at least the full version of the Motorola assembler and linker. Additionally, the Motorola C compiler is recommended.

EVM56xxx ISA Interface Card Technical Data

The ISA interface card for Motorola DSPEVM56xxx boards (Signalogic P/N SHHI56-ISA), which interfaces directly to the host port on either the DSP56303EVM or DSP56002EVM† boards, has the following features:

- host port of data transfer allows transfer rates more than 20x faster than standard serial port method used by EVM56xxx boards
- fast download of program and vector data to EVM56xxx boards from DSPower-Block Diagram environment, or from user-defined host PC programs using Visual Basic, MATLAB, or C/C++ programming interface provided by DSPower-HWLib
- 5 ft. 25-wire ribbon cable with active-termination adaptor at EVM-end; adaptor contains +5V and +12V host PC power LED indicators (note: since ribbon cable supplies all power requirements, EVM56xxx DC power supply usage is optional)
- connects to “ISA connector" on EVM5630x and to host port IDC header on EVM56002
- all EVM-end signals are protected with 150Ω termination, allowing “hot” connection and disconnection
- consecutive group of 16 host PC I/O addresses are consumed; I/O base address is DIP switch-selectable on 0x10 (10H) boundaries, from I/O address 0x200 to 0x3f0
- DSP interrupt to PC is jumper-selectable

Software Interface Details

Although the Hypersignal and DSPower software packages contain real-time DOS and Windows drivers for the EVM56xxx ISA interface card, the information below may be applicable for user-defined software implementations.

Host PC I/O Address	**ISA Interface Card Function**
xx0..xx7	direct read/write to DSP56xxx host port
xx8	write to control latch (CTRL register)

CTRL register definition:

Bit	**DSP56xxx Function**
0	INTMODE (selects polarity of the INT to PC)
1	MODC
2	MODB
3	MODA
4	RESET
5	HCS
6	INTENABLE (write 0)
7	DMAENABLE (write 0)

Suggested reset sequence for EVM56xxx boards (assuming default I/O base address of 0x340):

```
outp(0x348, 0x0e);      // reset on, MODA, MODB, MODC high
outp(0x348, 0x1e);      // reset off, MODA, MODB, MODC high
...
...                     // access DSP56xxx host port as needed
...
outp(0x348, 0);         // reset on, MODA, MODB, MODC low
                        // while EVM56xxx board is inactive
```

Physical and Power Specifications

Typ. Current	250 mA @ +12V, 50 mA @ +5V, fuse protected
Max. Current	900 mA @ +12V, 900 mA @ +5V
Size	5" length x 3.15" height, XT-style connector

† NOTE: For EVM56002 boards, a boot EPROM (supplied by Signalogic) must be installed, and a “blue-wire” connection must be made from the adaptor (EVM-end) to the EVM56002 reset switch.

SIGNALogic®

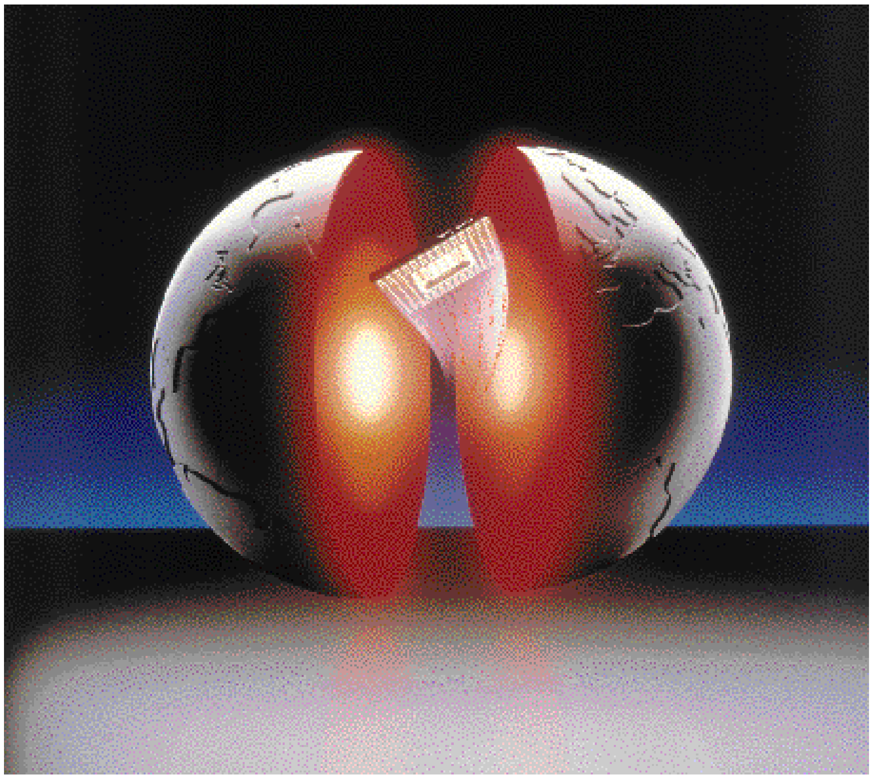
For Your DSP Answers, Call 1-800-DSPower®

Signalogic, Inc.
9617 Wendell @ Skillman • Dallas, TX 75243
tel: 214-343-0069 • fax: 214-343-0163
e-mail: dspinfo@signalogic.com • web: www.signalogic.com

TASKING

Quality Development Tools Worldwide

The Total Development Solution for the Motorola DSP56xxx Families



Motorola's family of digital signal processors offers a range of high-precision, high-performance 16 bit and 24 bit DSP's, targeted at data intensive, speed hungry applications, such as data communictions and multimedia computing environments and sound systems.

TASKING has designed an integrated toolset to use the features of the complete DSP56xxx families, to ensure that the power and flexibility of each processor is available to the development engineer. The TASKING toolset delivers an integrated environment consisting of powerful C and C++ compilers, CrossView debugger and our EDE.

C++, C Compiler

Efficient ANSI C compilers support the members of the DSP56xxx families

Assembler

Motorola compliant assembler with pipeline optimization

CrossView Debugger

C++, C and assembly debugging with OnCE™ /JTAG support

EDE

Integrated project environment with push button control over a variety of development tasks

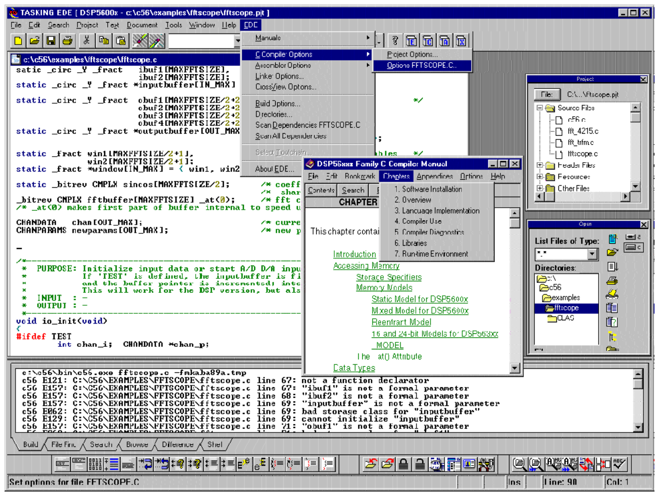
the DSP56xxx Toolset

The TASKING software development toolset for the DSP56xxx families provides a complete and cost-effective solution for programming this family of digital signal processors. The toolset supports a number of variants of this DSP family including the DSP56002, 56004, 56005, 56007, 56009, 56156, 56166 and the DSP56300, DSP56600 and DSP56800 cores.

the EDE

The Embedded Development Environment, voted EETimes 'Best Technology', integrates the members of the MIPS toolset to provide you with a single easy to use interface to all the tools needed for building, editing, compiling and debugging embedded applications. EDE is a language sensitive editor that recognizes both C and C++. It blends a command shell with powerful productivity tools and MS Windows™ resources. EDE gives you direct access to the tools and features you need to be your most productive. It delivers:

- Push-button control over a variety of development tasks spread over many tools
- Tight integration of tools enabling a rapid edit-compile-debug process, for maximum productivity
- Easy project setup including generation of makefiles
- Access to third party tools



```
c:\c56\examples\fftscope\fftscope.c
satic _circ _Y _fract   ibuf1[MAXFFTSIZE],
                        ibuf2[MAXFFTSIZE];
static _circ _Y _fract *inputbuffer[IN_MAX]

static _circ _Y _fract  obuf1[MAXFFTSIZE/2+2
                        obuf2[MAXFFTSIZE/2+2
                        obuf3[MAXFFTSIZE/2+2
                        obuf4[MAXFFTSIZE/2+2
static _circ _Y _fract *outputbuffer[OUT_MAX


static _fract win1[MAXFFTSIZE/2+1],
              win2[MAXFFTSIZE/2+1];
static _fract *window[IN_MAX] = { win1, win2

static _bitrev CMPLX sincos[MAXFFTSIZE/2];        /* coeff
                                                  /*  shar
_bitrev CMPLX fftbuffer[MAXFFTSIZE] _at(0);       /* fft c
/* _at(0) makes first part of buffer internal to speed u

CHANDATA   chan[OUT_MAX];                         /* curre
CHANPARAMS newparams[OUT_MAX];                    /* new p


_

/*--------------------------------------------------------
 *  PURPOSE: Initialize input data or start A/D D/A inpu
 *           If 'TEST' is defined, the inputbuffer is fi
 *           and the buffer pointer is incremented; inte
 *           This will work for the DSP version, but als
 *  INPUT  : -
 *  OUTPUT : -
 *--------------------------------------------------------
void io_init(void)
{
#ifdef TEST
        int chan_i;  CHANDATA *chan_p;
```

```
c:\c56\bin\c56.exe fftscope.c -fmkaba89a.tmp
c56 E121: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 67: not a function declarator
c56 E157: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 67: "ibuf1" is not a formal parameter
c56 E157: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 68: "ibuf2" is not a formal parameter
c56 E157: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 69: "inputbuffer" is not a formal parameter
c56 E062: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 69: bad storage class for "inputbuffer"
c56 E129: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 69: cannot initialize "inputbuffer"
c56 E157: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 71: "obuf1" is not a formal parameter
```

Project Management

EDE is more than a language sensitive editor. It is a complete environment which gives you direct access to the tools and the features you need in your projects. EDE lets you define the files of your project and, by navigating through the tabs, select the tool options that apply. With the EDE project manager you create and maintain a project so your application is always up-to-date. All aspects of a project are saved in the project file: the source files that make up the application, the tool options you selected (compiler, assembler, linker/locator and debugger), the tool directories and the options describing the building process. The project manager handles file dependencies as well as the exact sequence of operations to build you application.

When you push the make button EDE generates a complete makefile, including file dependencies, and builds your application. EDE interprets the error messages generated by the tools and shows you where the errors are, so you can fix them quickly. A simple click on the debug button launches the debugger directly from your project environment.

the C++ Compiler

TASKING delivers the power of object oriented design and coding techniques for the DSP56xxx families. The C++ language is a superset of the C language allowing the intermixing of C and C++ within a single application. Thus the benefits of C++ can be incorporated into an existing C application one module at a time, providing a graceful migration from C to C++.

The C++ compiler will be implemented as a front end that generates C together with debugging information so that you can debug using C++ objects. The C++ compiler will support the ANSI/ISO language standard, once it is approved. Until that time the compiler conforms to the language definition of *The Annotated C++ Reference Manual*. The compiler also has a compatibility mode with AT&T's C++ front end v 3.0.

The advantage of C++ is that it provides:

Powerful enhancements to C which increase the language's expressive power and reduce the likelihood of errors:
•Dynamic allocation of objects
•Passing references
•Operator overloading
•Default values
•Inline functions
•Rigorous type checking

Object Oriented Programming which increases productivity, improves quality and provides reusability:
•Encapsulation
•Data Hiding
•Inheritance

the C Compiler

The C56xxx compilers are designed and built specially for each member of the Motorola DSP family, to bring you the ultimate in code efficiency without violating the ANSI-C standard. To achieve this code efficiency the compiler uses a number of optimization techniques and supports a number of language extensions and inline functions:
•Full ANSI C compiler to ensure early error detection.
•Support for all DSP56xxx families
•Extensive optimizations for very efficient code
•Hardware loops using DO and REP.

- •MAC instruction in computational C expressions.
- •Two memory models, *static* and *reentrant*
- •Reentrant libraries
- •Fixed point arithmetic support with _fract data type
- •Single precision floating point
- •Complex data type
- •Built-in support for overflow/saturation.
- •Storage specifiers for X,Y,L and P memory
- •16 bit pointer arithmetic
- •Inline assembly
- •Fast parameter passing by avoiding the stack
- •Complete C and runtime libraries
- •DSP specific data types allow you to program your filters in C without noticeable overhead.

Compiler Optimization

The C compiler has been designed to generate very efficient code, close to the efficiency of hand coded assembly. In addition to the usual compiler optimizations, three particular optimization techniques were selected and applied extensively because they had the most significant effect on code efficiency:

Common subexpression elimination

The compiler detects expressions which occur repeatedly in an application. Keeping the intermediate result in a register boosts the performance of the code, since the next time the result is already available.

Loop recognition

When expressions inside a loop can be placed (partially) outside the loop execution time is drastically reduced.

Variable usage analysis and register allocation

The compiler schedules and allocates register resources such that maximum usage will be reached.

Effective use of specific Motorola DSP features, such as hardware do-loops, bit manipulating instructions and the multiply accumulate instruction guarantee excellent code quality. Other optimizations include:

- •constant folding
- •expression rearrangement
- •expression simplification
- •loop rotation
- •rewrite logical expression into conditional jumps
- •reduce conditional change flow
- •jump chaining
- •dead code elimination
- •remove useless jumps
- •conditional jump reversal
- •cross jumping & branch tail merging
- •constant & copy propagation
- •loop optimizations
- •pseudo register allocation
- •peephole optimizations
- •dead assignment elimination
- •leaf function handling
- •loop unrolling
- •dead storage elimination
- •tail recursion elimination
- •replacing NOP’s
- •instruction parallelization
- •hardware DO an REP loops
- •MAC instruction generation
- •absolute addressing mode usage

Efficiency

In addition to the optimizations that the compilers carry out automatically there are various other features that help you to optimize and tune your code:

- •Function prototyping gives you control over the size and number of parameters
- •A *static* and *reentrant* memory model allow you to choose how parameters are passed: using the stack or, more efficiently, via static, directly accessible memory
- •Inline expansion of predifined functions, such as *_abs, _bfchg, _bfclr, _bfset, _bftsth, _bftstl, _rol, _ror, _stop, _cmul, _cadd, _cdiv, _nop, _swi, _wait, _rnd, _cond_scale, _pdiv, _sqrt*
- •Adjustable code generation with *#pragma’s*
- • Circular buffer data type for efficient filter loops
- •Floating point libraries with limited exception trapping to reduce runtime argument checking

The user manual provides a number of recommendations on how to make the compiler produce the most efficient code.

Memory Models

The compilers support three memory models: reentrant, static and mixed reentrant/static model. The reentrant model uses the stack to allocate automatics and pass parameters. To augment the hardware stack of the DSP5600x a software stack is created to support the reentrant model. One of the drawbacks of this is that it uses relatively expensive addressing modes to access objects on the stack. Therefore the compiler also supports a static memory model. The compiler allocates space for automatics and parameters in a static memory area. To keep the size of this area limited, the linker overlays the static areas of the individual functions based on the call graph of the application: automatic variables and parameters of functions that never call each other (indirectly) may use the same memory space.

Pragmas

The *#pragma* directive supplies target dependent data to the compiler without violating the ANSI C language, making the resultant program portable. There are a number of pragmas supported by the C56xxx compilers many of which are common to all of our compilers. Pragmas are used to control optimization, to control data allocation for optimum performance, merge C lines with generated assembly, to control code generation for interrupt service routines and to support inline assembly programming.

Data Types and Sizes

All ANSI C types are supported. In addition to these types, a complex type, _fract (fixed point) and _circ (circular buffer) have been added.

Data Type	**DSP 561xx** size in bits	**DSP56xxx** size in bits
signed char	8	8
unsigned char	8	8
signed short	16	16
unsigned short	16	16
signed int	16	24
unsigned int	16	24
signed long	32	48
unsigned long	32	48
_fract	16	24
long _fract	32	48
pointer	16	16
float/double	32	32
enum	16	24
complex	32 (2 * 16)	48 (2 * 24)

Libraries

The compilers are delivered with libraries for all the different members of the DSP56xxx families. Each set consists of ANSI C libraries, run time libraries and fixed and floating point libraries.

Software floating point libraries are supplied in single precision. The types *float* and *double* are both implemented as single precision floating point. The single precision libraries are available with and without trapping. The libraries without trapping are faster but out of range values will not be flagged as an error.

The compilers support various calculus types such as: integral types for integer arithmetic, a *_fract* type qualifier for fixed point arithmetic and *float* and *double* for floating point calculus.

the Assembler

The assembler is an integral part of the toolset and is compliant with Motorola's CLAS assembler package for the DSP5600x and the DSP561xx families, for example nested and fragmented sections support is compatible with the CLAS assembler. In addition to this language definition the assembler supports a number of controls and pseudo instructions such as section handling to implement section overlaying to save memory. The assembler also supports various controls to steer listing file generation. The assembler applies extensive optimization techniques to ensure optimal instruction sequences:

Instruction parallelizing

The Motorola DSPs support parallel execution of move instructions with other instructions: If an assembler program contains a `move` instruction that can be executed in parallel with a preceding or succeeding instruction, the assembler replaces these two instructions with the combined equivalent, which executes faster on the processor and saves ROM space. For example:

```
and X0, A1
move (R5) -N5
```

will be replaced by:

```
and X0, A1   (R5) -N5
```

Nop insertion & delay slot filling

Due to the pipeline architecture of the DSP family certain sequences of instructions are not allowed, or can have unexpected behavior. Moving data from memory into a register takes several cycles to complete, so the first instruction following that move cannot use the new register contents. The assembler checks whether instruction sequences with pipeline hazards occur and attempts to avoid them by reorganizing the code. If no suitable instruction sequences are available then the assembler inserts a NOP instruction to force deterministic (and expected) behavior of the program.

Nop removal

If a program has been coded with to many NOPs, the assembler automatically removes the NOPs to save code space and increase program performance.

Branch Optimizations

Branch and jump instructions in the DSP5600x and DSP561xx cores are always absolute. Whether the assembler chooses a long or short instruction format depends on the memory location of the branch target. Selecting the long instruction format when the short form would have been adequate can become a costly waste of memory and performance. The assembler builds a complete flow graph of the source module and uses this flow information in combination with section allocation attributes of the branch target, to select the most efficient addressing modes.

Macro preprocessing

The assembler has a built-in macro preprocessor which features an include file mechanism, macro definition and expansion as well as conditional assembly.

Macro's can be defined and undefined at any place in the source. The preprocessor supports a set of controls which help you to create structured assembly programs. With controls like *.if*, *.while* and *.for* you can write clear structured assembly programs without writing virtually the same sequences of assembly instructions over and over again.

the Linker/Locator

The linker combines relocatable objects, generated by the assembler into a new combined relocatable output file. Modules will be extracted from object libraries if these modules resolve unsatisfied external references of one of the object modules. Libraries are scanned until they no longer resolve external references. The linker output file is then processed by the locator to assign absolute memory locations to the application. Features include:

- •Overlaying linker to reduce memory space
- •Partial linking
- •Incremental linking
- •Linker map file to help with debugging
- •Flexible locator control language to control memory layout of your application
- •Default control file for locator covers most applica tions
- •Automatic inclusion of library modules

The linker uses two specific techniques to improve program quality:

Overlaying sections

If a program contains sections that can be overlayed then the linker builds a call graph (a tree structure which lists the references in the program). Data accessed by code sections which cannot be active at the same time may be allocated at the same addresses. This makes the program more memory efficient.

Global type checking

The C compiler checks whether function definitions and function references match; the function return type and the parameter types have to be defined and used consistently. Since the compiler processes only one C module at a time, mismatches of declarations cannot be detected if they occur in different modules. The linker processes all modules of the application and checks whether all function parameters and return types in your entire program are consistent. In this way mismatches of data declarations across module boundaries are detected by the linker.

Locator control language

The purpose of the locator is to assign memory to an application. Relocatable sections do not have absolute addresses assigned until the locator phase. Before then they, may be placed anywhere in memory. This allows you to write your program independent of its place in memory. This improves the maintainability of your program and makes it easier to extend in the future. The tool set also supports absolute sections by defining the absolute memory addresses explicitly in the source file.

The locator assigns absolute addresses to a program using a locator control file. This file is written in a C like language called DELFEE (DEscriptive Language For Embedded Environments). The locator description file describes the characteristics of your processor, the layout of physical memory of your target board and the assignment of the software components in your program to this memory. Template locator description files and automatic assignment of default locator settings ensures ease of use (Invocation of the locator without explicit inclusion of a control file, automatically forces the locator to use its defaults).

The flexibility of the DELFEE language guarantees that the locator supports any memory configuration imaginable and checks for the most suitable assignment and allocation of the software components.

Overlaying memory banks

The locator description file allows you to allocate code and data at the same logical addresses, whereas the physical allocation resides in different memory banks, this is called overlaying. The TASKING assembler package supports two types of overlaying:

Linker Overlaying

The linker maps sections that may share the same address space at the same logical addresses (the addresses used in the program to access memory). The result is a new single section.

Locator Overlaying

The locator allows you to allocate several sections in the same logical memory space, though they have their own (different) physical allocation in memory on the target.

Locator output format

The locator supports various object formats: IEEE-695 for symbolic debugging with CrossView, Motorola-S records, Intel Hex records and Motorola CLAS compatible object format.

the Debugger CrossView

CrossView is our high level language debugger designed to deliver functionality that will reduce the time spent testing and debugging. It combines the flexibility of the C language with the control of code execution found in assembly language debugging. CrossView brings the full power of Microsoft and X Windows to the debugging environment by displaying and updating the most critical execution data in an organized way.

CrossView for the the Motorola DSP family supports the on-chip debugging OnCE™ (On-Chip Emulation) available on all the family members.

Productive Debugging

CrossView has a number of productive debugging features designed specially for the embedded systems programmer:

•Single Stepping allows you to watch a program execute line by line. You can step single lines of source or, step in or over procedure calls.

•Code in C, Debug in C. Automatic correlation of variables and line numbers.

•Stack Trace shows the program's function calling path. Function calling sequence and parameter values passed are displayed at each level.

•Databreakpoints are supported through the OnCE facility. You can set a break condition on a read or write operation on a specific, or range of addresses.

•Code breakpoints let you halt the program in critical places and observe values. Code breakpoints can be permanent, temporary or conditional by specifying standard C conditions.

•Assertions let you execute user specified command lists after running every line of source code. This is the software equivalent of data breakpoints, they can be used to set up sophisticated error checking mechanisms that uncover the most elusive bug.

•Simulated I/O enables you to debug without input/ output devices being available. Screen, keyboard or files can be used as I/O devices and eight streams can be active at a time.

•Direct Register and Memory Access via the OnCE interface on the processor.

•OnCE Trace Window shows the contents of the OnCE trace buffer

•Internal Instrucion Counter allows you to monitor how many instructions have been executed for timing purposes.

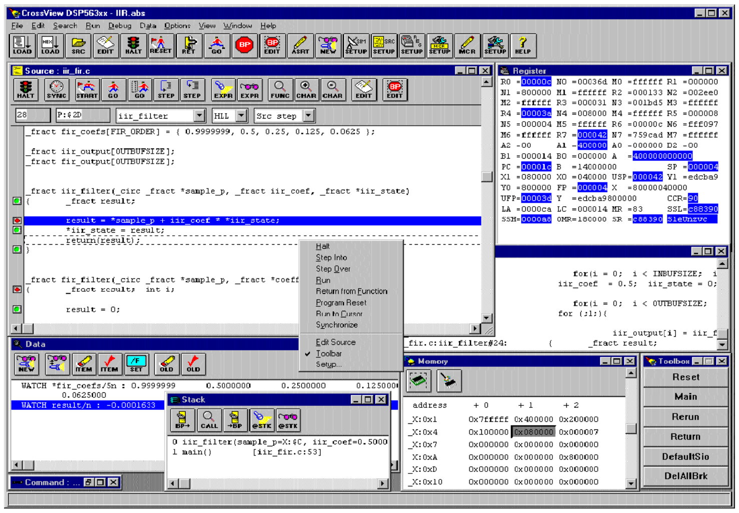
•Data Monitoring is used to monitor any expression or variable continually.

•C Expression Evaluation lets you enter C expressions or CrossView commands in any combination and have them evaluated.

•Macros let you store and recall complex commands and expressions via buttons.

Multi-windowInterface

CrossView brings the full power of Microsoft and X Windows to organize and display every facet of the application as it is running in the target system. CrossView features multiple child windows, extensive menus and dialog boxes and user programmable buttons for macros. CrossView also has an accelerator bar which provides quick access to frequently used debugging commands.



```
_fract fir_coefs[FIR_ORDER] = { 0.9999999, 0.5, 0.25, 0.125, 0.0625 };

_fract iir_output[OUTBUFSIZE];
_fract fir_output[OUTBUFSIZE];


_fract iir_filter(_circ _fract *sample_p, _fract iir_coef, _fract *iir_state)
{        _fract result;

         result = *sample_p + iir_coef * *iir_state;
         *iir_state = result;
         return(result);
}


_fract fir_filter(_circ _fract *sample_p, _fract *coeff
{        _fract result;  int i;

         result = 0;
```

```
R0 =00000c N0 =00036d M0 =ffffff R1 =000000
N1 =800000 M1 =ffffff R2 =000133 N2 =002ee0
M2 =ffffff R3 =000031 N3 =001bd5 M3 =ffffff
R4 =00003a N4 =008000 M4 =ffffff R5 =000008
N5 =000004 M5 =ffffff R6 =00000c M6 =fff097
M6 =ffffff R7 =000042 N7 =759cad M7 =ffffff
A2 -00     A1 -400000 A0 -000000 D2 -00
B1 =000014 B0 =000000 A  =400000000000
PC =00001c B  =14000000          SP =000004
X1 =080000 X0 =040000 USP=000042 Y1 =edcba9
Y0 =800000 FP =000004 X  =80000040000
UFP=00003d Y  =edcba9800000      CCR=90
LA =0000ca LC =000014 MR =83     SSL=c88390
SSH=0000a8 OMR=180000 SR =c88390 SleUnzvc
```

```
        for(i = 0;  i < INBUFSIZE;  i
    iir_coef  = 0.5;  iir_state = 0;

        for(i = 0;  i < OUTBUFSIZE;
    for (;1;){

                iir_output[i] = iir_f
_fir.c:iir_filter#24:        {       _fract result;
```

```
WATCH *fir_coefs/5n : 0.9999999      0.5000000      0.2500000      0.125000
        0.0625000
WATCH result/n : -0.0001633
```

```
0 iir_filter(sample_p=X:$C, iir_coef=0.5000
1 main()          [iir_fir.c:53]
```

```
address       + 0       + 1       + 2
_X:0x1      0x7fffff  0x400000  0x200000
_X:0x4      0x100000  0x080000  0x000007
_X:0x7      0x000000  0x000000  0x000000
 X:0xA      0x000000  0x000000  0x800000
_X:0xD      0x000000  0x000000  0x000000
_X:0x10     0x000000  0x000000  0x000000
```

Windows available include:
- Command Window
- Stack Window
- Source Window
- Simulated I/O Window
- Data Window
- Breakpoint Window
- Register Window
- Help Window
- Macro Window
- Trace window

User Manuals

Context Sensitive Help provides complete command syntax and detailed descriptions with hypertext links to the user manual.

Target Environment

CrossView supports Motorola's popular Evaluation Module (EVM) and ADS (Application Development System) execution environment for the DSP56xxx families. ADS includes a target board with the target DSP, a host board, to ensure optimal communication between the host and the target board, and an OnCE cable.

For more information (about CrossView) see the CrossView datasheet.

Availability

The tools to support the DSP5600x, DSP561xx and DSP56300 core are available now. Call us for availability of the DSP66600 and DSP56800 cores. All tools for the DSP56xxx families will be available to run on:

PC	Windows, Windows 95 and Windows NT
SUN	SUNOS and Solaris
HP9000	HPUX

Customer Support

Purchasing TASKING products marks the beginning of a long term relationship. TASKING is dedicated to providing quality products and support worldwide. This support includes program quality control, product update service and support personnel to answer questions by telephone, fax or email.

TASKING product support begins before the purchase of any of our products with extensive testing in order to ensure our high standard of quality assurance.

A 90 day maintenance plan is included with the purchase of TASKING products and entitles the customer to enhancements and improvements as well as individual response to problems. Annual maintenance contracts are available at the end of the 90 day maintenance plan. This extremely valuable service, in return for a small annual fee, provides the user with all program enhancements released during the period of the program maintenance agreement, and assures response to all problem reports submitted by the user.

Our email address in the US is: support_us@tasking.com
Our email address in Europe is: support_nl@tasking.com

TASKING

Unparalleled experience in embedded applications

Since pioneering the concept of “cross development” over 20 years ago, Tasking has been the industry leader in the development, manufacture, and support of software development tools for DOS, Unix, and VMS programming environments.

With 100+ employees, annual revenues over $15 million, and a global distribution network in North America, Europe, and Asia, we are the leading supplier of software development tools for embedded microcontroller and Digital Signal Processors applications across industry standard computing platforms. These include products supported on the PC, Sun, Sparc, HP9000 DECStation and DEC Alpha.

Commitment to a total development solution

Tasking is committed to providing you with highly integrated, leading-edge software tools for embedded development, across industry-standard computing platforms, coupled with the most comprehensive support and training services available.

To complete the development solution, we work closely with a number of vendors to ensure that our tools work with all the tools you need. We co-operate with hardware manufacturers for emulators, logic analysers and evaluation boards and software manufacturers for real time kernels, device drivers, VHDL simulators and more.

For more information about us and the microprocessors we support, visit our web site at www.tasking.com

Offices are located at:

UNITED STATES (International Headquarters)
TASKING
Norfolk Place, 333 Elm Street
Dedham, MA 02026-4530. United States
phone: 1-800-458-8276 outside US 617-320-9400
fax: 617-320-9212
email: sales_us@tasking.com

THE NETHERLANDS (European Headquarters)
TASKING Software BV
P O Box 899
3800 AW Amersfoort. The Netherlands
phone: +31 33 4 558584 fax: +31 33 4 550033
email: sales_nl@tasking.com

GERMANY
TASKING Software Deutschland GmbH
Brennerstraße 5
71229 Leonberg. Germany
phone: +49 7152 97991 0 fax: +49 7152 97991 20
email: sales_de@tasking.com

ITALY
TASKING Software Italia Srl
Via Napo Torriani 29
20124 Milano. Italy
phone: +39 2 6698 2207 fax: +39 2 6698 2189
email: sales_it@tasking.com

JAPAN
Nihon TASKING
Shiba-ST Building, 6th Floor
1-15-31 Shiba
Minato-ku
Tokyo 105. Japan
phone: +81 3 3457 6831 fax: +81 3 3457 6834
email: sales_jp@tasking.com

UNITED KINGDOM
TASKING Ltd
Enterprise House, Ocean Village
Southampton, Hants SO14 3XB
United Kingdom
phone: +44 1703 334774 fax: +44 1703 334772
email: sales_uk@tasking.com

INTERNET
http://www.tasking.com

Your DISTRIBUTOR

TASKING
Quality Development Tools Worldwide

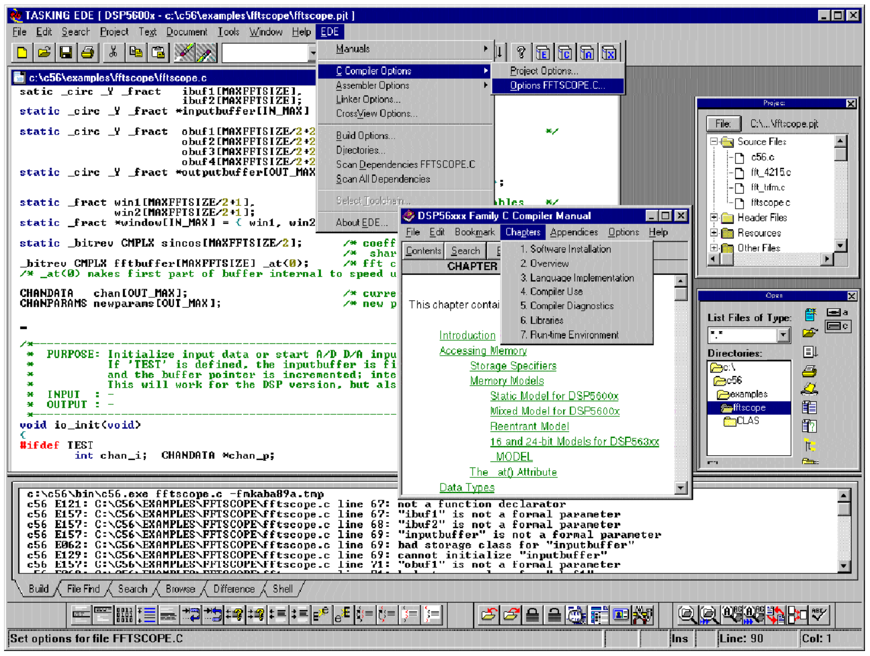
TASKING
Quality Development Tools Worldwide

Integrated Tools at Work

The Total C and C++ Development Solution for the Motorola DSP563xx Family

Toolset Features

- ✓ Integrated project environment
- ✓ Dedicated DSP563xx compiler
- ✓ Upward compatible with TASKING DSP5600x tools
- ✓ Supports 16 bit mode and 24 bit modes of DSP563xx cores
- ✓ Compiler generated DO-REP loops
- ✓ Effective use of DO-FOREVER and BRKcc in loops
- ✓ Efficient stack handling; no software stack required
- ✓ Built-in functions and pragma's for cache handling
- ✓ Packed strings and character arrays for efficient memory use
- ✓ Tight code generation for bit field operations, using EXTRACT, EXTRACTU and INSERT instructions
- ✓ Optimizing assembler that rearranges instructions for parallel execution and branch optimization
- ✓ Linker with cross-module type checking for C functions and variable declarations
- ✓ CrossView debugger, interfacing to Motorola EVM and ADS via the ONCE™ port



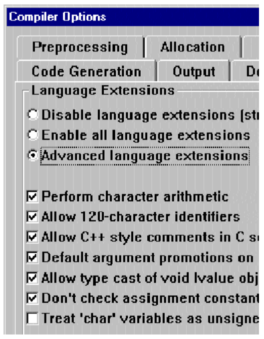
c:\c56\examples\fftscope\fftscope.c

```
satic _circ _Y _fract    ibuf1[MAXFFTSIZE],
                         ibuf2[MAXFFTSIZE];
static _circ _Y _fract *inputbuffer[IN_MAX]

static _circ _Y _fract   obuf1[MAXFFTSIZE/2+2
                         obuf2[MAXFFTSIZE/2+2
                         obuf3[MAXFFTSIZE/2+2
                         obuf4[MAXFFTSIZE/2+2
static _circ _Y _fract *outputbuffer[OUT_MAX


static _fract win1[MAXFFTSIZE/2+1],
              win2[MAXFFTSIZE/2+1];
static _fract *window[IN_MAX] = { win1, win2

static _bitrev CMPLX sincos[MAXFFTSIZE/2];        /* coeff
                                                  /*  shar
_bitrev CMPLX fftbuffer[MAXFFTSIZE] _at(0);       /* fft c
/* _at(0) makes first part of buffer internal to speed u

CHANDATA   chan[OUT_MAX];                         /* curre
CHANPARAMS newparams[OUT_MAX];                    /* new p


/*---------------------------------------------------------
 *  PURPOSE: Initialize input data or start A/D D/A inpu
 *           If 'TEST' is defined, the inputbuffer is fi
 *           and the buffer pointer is incremented; inte
 *           This will work for the DSP version, but als
 *  INPUT  : -
 *  OUTPUT : -
 *---------------------------------------------------------
void io_init(void)
{
#ifdef TEST
        int chan_i;  CHANDATA *chan_p;
```

```
c:\c56\bin\c56.exe fftscope.c -fmkaba89a.tmp
c56 E121: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 67: not a function declarator
c56 E157: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 67: "ibuf1" is not a formal parameter
c56 E157: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 68: "ibuf2" is not a formal parameter
c56 E157: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 69: "inputbuffer" is not a formal parameter
c56 E062: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 69: bad storage class for "inputbuffer"
c56 E129: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 69: cannot initialize "inputbuffer"
c56 E157: C:\C56\EXAMPLES\FFTSCOPE\fftscope.c line 71: "obuf1" is not a formal parameter
```

Set options for file FFTSCOPE.C | Ins | Line: 90 | Col: 1

- On-line manuals
- Librarian
- Object code reporter
- Cross reference reporter
- Built-in grep and file difference utility



Language Extensions
- ( ) Disable language extensions (st
- ( ) Enable all language extensions
- (●) Advanced language extensions

- ☑ Perform character arithmetic
- ☑ Allow 120-character identifiers
- ☑ Allow C++ style comments in C s
- ☑ Default argument promotions on
- ☑ Allow type cast of void lvalue obj
- ☑ Don't check assignment constant
- ☐ Treat 'char' variables as unsigne

Integrated Environment

TASKING's Embedded Development Environment, EDE, is an integrated environment for tools to create, edit, compile and debug your application. EDE lets you define the files of your project and, by navigating through the tabs select the TAWSKING tool options that apply. The EDE project manager provides a method of creating and maintaining a project so that your application is always up to date. All aspects of a project are saved in the project file: the source files that make up the application, the tool options (compiler and CrossView debugger), the tool directories and the options describing the building process. The project manager handles file dependencies as well as the exact sequence of operations required to build your application.

EDE includes the following components:

- Tool option selection
- Automated make facility

EDE is more than a language sensitive editor, it is a complete environment which gives you direct access to the tools and features you need to be your most productive. EDE integrates error message output from the TASKING tools into your editing environment. You can browse from error to error with the ERR button while fixing your code.

EDE lets you define the files of your project and when you push the 'Make' button, EDE generates the complete makefile, including all dependencies, and builds your application. EDE understands the error messages generated by the tools and shows you where the errors are, so you can fix them quickly. Using Version Control is easy. EDE has an interface for checking in your changes, checking out a file for review or locking a revision you plan to change. You can use the default commands, specify your own, or interface to standard version control packages.

the C++ Compiler

The TASKING C++ Compiler for the DSP563xx Family delivers the advantages of object oriented programming to your DSP application development. The C++ compiler is implemented as a front end that generates C code. It will support the ANSI/ISO language standard, once this is approved. Until that time the compiler implements the language as defined in the Annotated C++ Reference Manual. Call us for availability.

the C Compiler

The C compiler is dedicated to the DSP563xx architecture. Various language extensions combined with powerful optimizations extend the applicability of the C language in DSP programs. These extensions include memory specifiers for X,Y, L and P memory , a fractional data type, a circular memory buffer type and a complex data type. You can select code generation for the 16 bit mode and the 24 bit mode of the DSP. Built-in functions give you direct access to the cache in your C code.

the Assembler

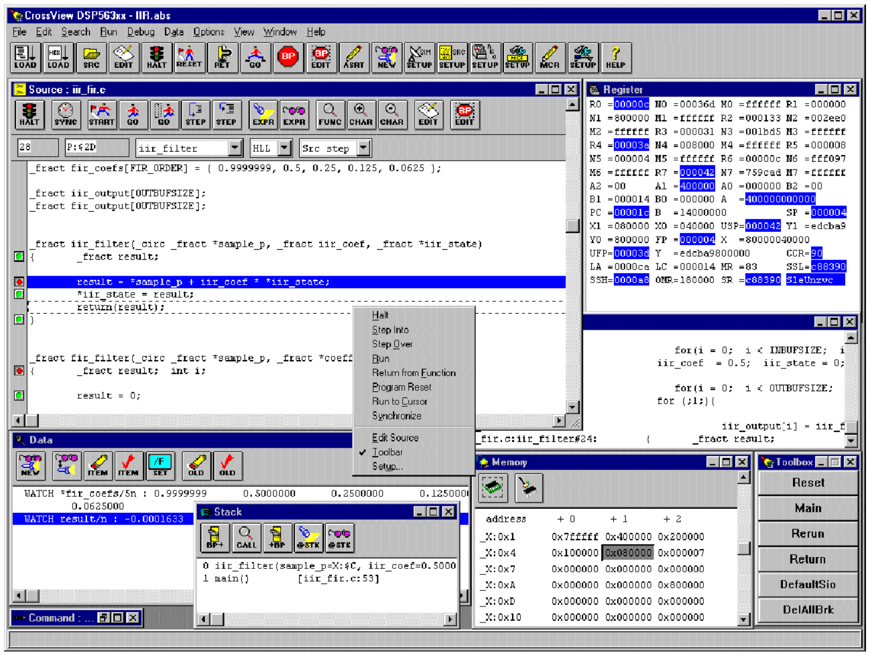
A powerful pipeline optimizer reorganizes your assembler code to find instructions which can execute in parallel. The assembler optimizes generic branches to the shortest branch possible. Automatic nop insertion and deletion eases program development, reducing the need to trace resource conflicts while writing code.

CrossView Debugger

CrossView for the DSP563xx Family interfaces to Motorola's Evaluation Module (EVM) and Application Development System (ADS). Various windows present the actual program status in an orderly manner. The source window shows your program as C source text, assembly source or a mixture of both. You can single step your program at C source line level or at instruction level. A highlighted register in the register window indicates the values changed in the previous execution step. The trace window displays the contents of the OnCE™/ϑTAΓ port trace buffer of the DSP563xx. The stack window shows the program stack, including function parameters. A configureable toolbox lets you define macros of you most favorite debugger commands. A record and playback facility allows non-interactive use of the debugger for regression tests on your program code.

Customer Support

TASKING is committed to provide quality products and support worldwide. This support includes program quality control, product update service and support personnel to answer questions by telephone, fax or E-mail.
A 90 day maintenance plan is included with the purchase of TASKING products and entitles you to product enhancements and improvements With the individual help of our support staff you can reduce development costs when you have questions how to use our tools effectively in your


CrossView DSP563xx - IIR.abs
File Edit Search Run Debug Data Options View Window Help
LOAD LOAD SRC EDIT HALT RESET RET GO BP EDIT ASRT NEW SETUP SETUP SETUP SETUP MCR SETUP HELP
Source : iir_fir.c
HALT SYNC START GO GO STEP STEP EXPR EXPR FUNC CHAR CHAR EDIT EDIT
28 P:$2D iir_filter HLL Src step
Halt
Step Into
Step Over
Run
Return from Function
Program Reset
Run to Cursor
Synchronize
Edit Source
✓ Toolbar
Setup...
Register
Data
NEW ITEM ITEM SET OLD OLD
Stack
BP+ CALL +BP @STK @STK
Memory
Toolbox
Reset
Main
Rerun
Return
DefaultSio
DelAllBrk
Command : ...

```
_fract fir_coefs[FIR_ORDER] = { 0.9999999, 0.5, 0.25, 0.125, 0.0625 };

_fract iir_output[OUTBUFSIZE];
_fract fir_output[OUTBUFSIZE];


_fract iir_filter(_circ _fract *sample_p, _fract iir_coef, _fract *iir_state)
{         _fract result;

          result = *sample_p + iir_coef * *iir_state;
          *iir_state = result;
          return(result);
}


_fract fir_filter(_circ _fract *sample_p, _fract *coeff
{         _fract result;  int i;

          result = 0;
```

```
R0 =00000c N0 =00036d M0 =ffffff R1 =000000
N1 =800000 M1 =ffffff R2 =000133 N2 =002ee0
M2 =ffffff R3 =000031 N3 =001bd5 M3 =ffffff
R4 =00003a N4 =008000 M4 =ffffff R5 =000008
N5 =000004 M5 =ffffff R6 =00000c N6 =fff097
M6 =ffffff R7 =000042 N7 =759cad M7 =ffffff
A2 =00     A1 =400000 A0 =000000 B2 =00
B1 =000014 B0 =000000 A  =400000000000
PC =00001c B  =14000000          SP =000004
X1 =080000 X0 =040000 USP=000042 Y1 =edcba9
Y0 =800000 FP =000004 X  =80000040000
UFP=00003d Y  =edcba9800000      CCR=90
LA =0000ca LC =000014 MR =83     SSL=c88390
SSH=0000a8 OMR=180000 SR =c88390 SleUnzvc
```

```
      for(i = 0;  i < INBUFSIZE;  i
   iir_coef  = 0.5;  iir_state = 0;

      for(i = 0;  i < OUTBUFSIZE;
   for (;1;){

                  iir_output[i] = iir_f
_fir.c:iir_filter#24:      {       _fract result;
```

```
WATCH *fir_coefs/5n : 0.9999999      0.5000000      0.2500000      0.1250000
        0.0625000
WATCH result/n : -0.0001633
```

```
0 iir_filter(sample_p=X:$C, iir_coef=0.5000
1 main()          [iir_fir.c:53]
```

```
address     + 0      + 1      + 2
_X:0x1     0x7fffff 0x400000 0x200000
_X:0x4     0x100000 0x080000 0x000007
_X:0x7     0x000000 0x000000 0x000000
_X:0xA     0x000000 0x000000 0x800000
_X:0xD     0x000000 0x000000 0x000000
_X:0x10    0x000000 0x000000 0x000000
```

specific situation.

Availability

The DSP563xx development tools are available to run on:
PC (Windows, Windows 95 and NT), SUN SPARC, HP9000.
The EDE development environment is available on PC only.

TASKING Offices:

UNITED STATES (International Headquarters)

phone: 1-800-458-8276   outside US: 617 320 9400
fax: 617-320-9212

THE NETHERLANDS (European Headquarters)

phone: +31 33 455 8584   fax: +31 33 455 0033

GERMANY

phone: +49 7152 97991 0   fax: +49 7152 97991 20

ITALY

phone: +39 2 6698 2207   fax: +39 2 6698 2189

JAPAN

phone: +81 3 3457 6831   fax: +81 3 3457 6834

UNITED KINGDOM

phone: +44 1703 334774   fax: +44 1703 3347

Internet:

http://www.tasking.com

563xxspd.pm5-0397-2

Product Note
March 1997

COSSAP™ Motorola DSP Developer Kits

A mixed-level hardware-software verification environment to support the debugging, analyzing, profiling, and optimization of firmware on embedded Motorola DSPs within the COSSAP DSP design environment.

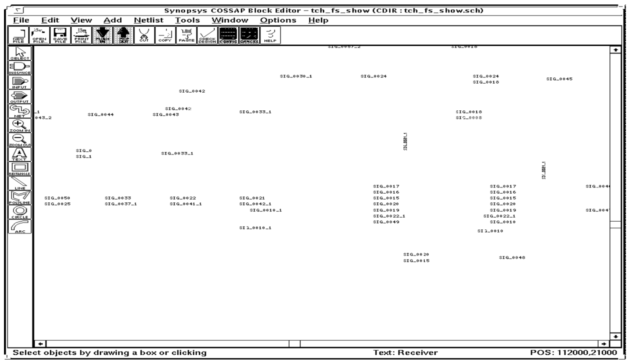
DESCRIPTION

Today's digital signal processing systems consist of many processing functions mapped onto software-programmable DSP cores and embedded logic. In many cases, it is extremely difficult to verify, debug, and profile the software implementation using traditional software development tools, especially if the software testing strongly depends on the system environment.

The COSSAP Motorola DSP Developer Kits enable designers of digital consumer and communication systems to incorporate models of Motorola DSP cores within their DSP system-level design environments. System designers working within the COSSAP design environment can conceive of algorithms and map them on to architectures with Motorola cores and embedded logic, and develop, debug, profile, and analyze the firmware. Working within the COSSAP environment, designers can architect complete DSP systems like modems and model real-world phenomenon, such as wireless channels, cable-TV channels, and satellite channels. This allows designers to verify the functionality as well as the MIPS performance of the DSP firmware. In addition, COSSAP allows true hardware-software cosimulation capabilities supporting all Motorola processors and industry-standard VHDL and Verilog HDL simulators.

PROCESSORS SUPPORTED

- DSP56000 Family: 56000, 56001, 56002, 56004, 56005, 56007, 56009, 56011
- DSP56100 Family: 56116, 56156, 56166
- DSP56300 Family: 56300, 56301, 56302, 56303, 56305
- DSP56600 Family: 56602, 56603
- DSP56800 Family: 56800, 56811



Synopsys COSSAP Block Editor - tch_fs_show (CDIR : tch_fs_show.sch)

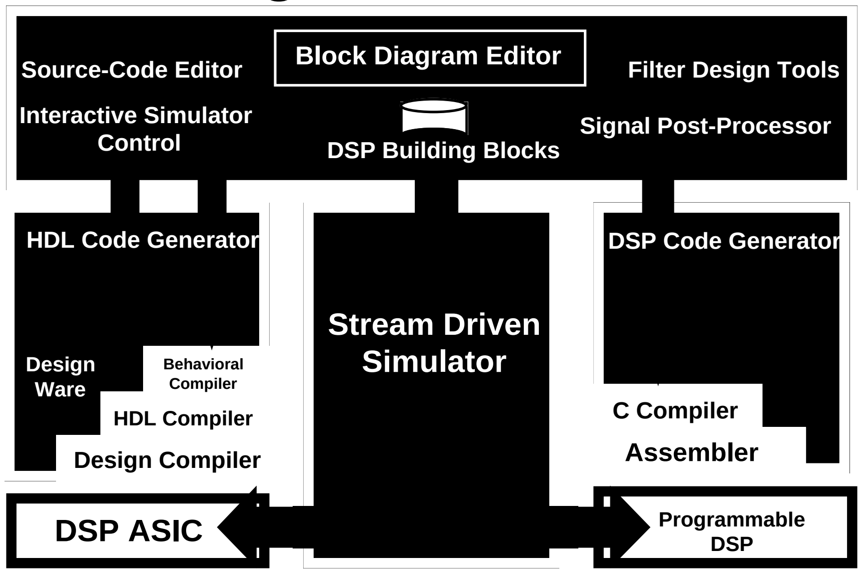
Select objects by drawing a box or clicking   Text: Receiver   POS: 112000,21000

USAGE

The COSSAP DSP Developer Kits (DDK) let you execute and debug assembly code programs in a system-level simulation environment. The assembly code blocks are executed at the instruction level by using an instruction-set simulator cosimulating with the COSSAP Stream Driven Simulator.

Each COSSAP DSP Developer Kit comes with a library of blocks reflecting various I/O config-urations of the DSP. Designers can include these DDK blocks in the system-level block diagram of the digital signal processing system and load an executable software program into them. Each DDK block represents a fragment of the software program, making it possible to develop highly modular software. The COSSAP block diagram creates a virtual interconnect between different software fragments, and the COSSAP Stream Driven Simulator coordinates the execution of single software fragments. During simulation, designers have full debugging support for both the instruction-set simulator and the COSSAP Stream Driven Simulator. Automatic synchronization between simulators is accomplished using dataflow techhniques. This allows each simulator to perform at a maximum level with minimum overhead.

The COSSAP Digital Communications Design Environment



PRODUCT DESCRIPTION

COSSAP is a complete digital signal processing (DSP) design tool suite that is used by designers of DSP systems to create, explore, and test algorithms, architectures, and their implementations for a variety of DSP applications such as speech coding, wireless transmission, data- and voice-band modem design, data storage, and image processing.

These algorithms are comprehensively specified by algorithm developers using the COSSAP block-diagram editor. These block diagrams are used as input specifications for simulation and implementation. Extensive parameterized simulations are performed using the Stream-Driven Simulator™, a dataflow simulator. These features are used by algorithm developers to verify the performance and behavior of the algorithm(s).

Implementation on programmable DSPs is done in COSSAP by DSP system and firmware designers through DSP assembly-code generation, where C/assembly code is generated for a target processor. Implementation onto hardwired DSP architectures is done by designers first exploring the architecture space, and taking the optimum architecture to gates. This is done in COSSAP via Behavioral and RTL HDL code generation, and then using Synopsys' Behavioral Compiler for architectural exploration, and Design Compiler for logic synthesis.

Verification of these implementations is done at the system level by DSP system and hardware designers through co-simulation of the implementation-specific representation (DSP assembly code or HDL) with the system-level block diagram specification.

KEY FEATURES

•Over 1200 library models to build up signal processing and communication systems
•Reference Design Kits for CDMA, TDMA, and GSM
•Stream-driven data-flow simulator kernel offers fast algorithm design cycles
•Ability to easily model and simulate asynchronous and multirate systems
•Hardware implementation path through VHDL and Verilog HDL Code Generation
•Software implementation path through DSP Code Generation and processor-specific compilers
•HW/SW verification through co-simulation with HDL Simulators and Instruction-Set Simulators

SYNOPSYS®