

Spherex, a subsidiary of Audio Products International Corp. (API), designed the Xbox 5.1 system based on TI's TMS320DA601 audio DSP. DA601, a member of the Aureus family, is a 225MHz floating-point processor that delivers 1800 million instructions per second and 32/64-bit native processing.

## Description

The TMS320C67x DSPs are the floating-point DSP family in the TMS320C6000 TMS320C6701 (C6701) device is based on the high-performance, advanced VelociTI very-long-instruction-word (VLIW) architecture developed by Texas Instruments (TI), making this DSP an excellent choice for multichannel and multifunction applications. With performance of up to 1 giga floating-point operations per second (GFLOPS) at a clock rate of 167 MHz, the C6701 offers cost-effective solutions to high-performance DSP programming challenges. The C6701 DSP possesses the operational flexibility of high-speed controllers and the numerical capability of array processors. This processor has 32 general-purpose registers of 32-bit word length and eight highly independent functional units. The eight functional units provide four floating-/fixed-point ALUs, two fixed-point ALUs, and two floating-/fixed-point multipliers. The C6701 can produce two multiply-accumulates (MACs) per cycle for a total of 334 million MACs per second (MMACS). The C6701 DSP also has application-specific hardware logic, on-chip memory, and additional on-chip peripherals.

The C6701 includes a large bank of on-chip memory and has a powerful and diverse set of peripherals. Program memory consists of a 64K-byte block that is user-configurable as cache or memory-mapped program space. Data memory consists of two 32K-byte blocks of RAM. The peripheral set includes two multichannel buffered serial ports (McBSPs), two general-purpose timers, a host-port interface (HPI), and a glueless external memory interface (EMIF) capable of interfacing to SDRAM or SBRAM and asynchronous peripherals.

The C6701 has a complete set of development tools which includes: a new C compiler, an assembly optimizer to simplify programming and scheduling, and a Windows source code execution.

# AUREUS™ TMS320DA610, TMS320DA601 FLOATING\_POINT DIGITAL SIGNAL PROCESSORS

*SPRS002H - SEPTEMBER 2001 - REVISED JANUARY 2004*

- Aureus™ High-Performance 32-/64-Bit Audio Digital Signal Processors (DSPs)
- DA610-250 MHz, 2000 MIPS/1500 MFLOPS
- DA601-225 MHz, 1800 MIPS/1350 MFLOPS
- Single DSP Solutions for Multichannel Audio Applications: A/V and DVD Receivers, Multi-Zone Receivers, High Speed Encoder, Simultaneous Encode/Decode, Surround Headphone, Speaker Virtualization, Room Correction
- DA601 and DA610 Compatibility Provides a Scalable Audio Solution Based on a Single DSP Instruction Set Architecture
- Certified Algorithms:
  - Dolby Digital Decoder, Dolby Digital EX,
  - Dolby Pro Logic IIX, Dolby Pro Logic II
  - DTS 5.1, DTS-ES 6.1, DTS Neo:6,

- DTS 96/24
- MPEG-2 AAC
- THX Ultra 2
- Supports Other Algorithms Including:
  - MP3 CODEC
  - WMA CODEC
  - SRS Circle Surround II
  - Waves' MaxxBass Technology
- Highly Optimized C/C++ Compiler
- VelociTI™ Advanced Very Long Instruction Word (VLIW) C67x™ DSP Core
  - Native Instruction Set Support for:
    - 32-/64-Bit IEEE 754 Floating-Point
    - 32/40/64 & Packed-16-Bit Fixed-Point
  - Eight Independent Functional Units:
    - Two ALUs (Fixed-Point)
    - Four ALUs (Floating- and Fixed-Point)
    - Two Multipliers (Floating-/Fixed-Point)
  - Fast Time to Market with RISC-Like ISA
- Low-Cost, Two-Level Memory System
  - 4K-Byte L1P Plus 4K-Byte L1D Cache
  - 256K-Byte L2 Cache/RAM (DA610)
  - 128K-Byte L2 Cache/RAM (DA601)
  - 512K-Byte L2 ROM (DA610/DA601)
- 32-Bit External Memory Interface (EMIF) Seamlessly Expands Memory Space by Supporting: SRAM, SDRAM, FLASH, EPROM, and SBSRAM. Four External Address Spaces
- 16-Bit Host-Port Interface (HPI) Enables High-Speed Encode/Decode Applications
- Enhanced Direct-Memory-Access (EDMA) Controller With 16 Independent Channels
- Two Multichannel Audio Serial Ports (McASPs)
  - Independent Dual Zone Audio on a Single DSP
  - 16 Data Pins (32 Channel Stereo)
  - Flexible Clocking
  - TDM Streams 2-32 Channels per Pin
  - Data Formatting Unit Supports Wide Variety of I2S and Similar Formats
  - Integrated Digital Audio Interface Transmitter (DIT) With Enhanced Channel Status/User Data
  - Extensive Error Checking and Recovery
- Two Inter-Integrated Circuit Bus (I2C Bus™) Multi-Master and Slave Interfaces
- Two Multichannel Buffered Serial Ports (McBSPs) Supporting:
  - Serial-Peripheral-Interface (SPI)
  - High-Speed TDM Interface
- Two 32-Bit General-Purpose Timers
- Two General-Purpose Input/Output Modules

- On-Chip Oscillator and PLL Module
- IEEE-1149.1 (JTAG†) Boundary-Scan-Compatible
- Package Options:
  - 208-Pin PowerPAD™ PQFP (suffix PYP)
  - 272-Pin Ball Grid Array (suffix GDP)
- 0.13- $\mu$ m Copper Metal CMOS Process
- 3.3-V I/Os, 1.2-V Internal

Aureus, VelociTI, C67x, and PowerPAD are trademarks of Texas Instruments. All trademarks are the property of their respective owners.

† IEEE Standard 1149.1-1990 Standard-Test-Access Port and Boundary Scan Architecture.

# Specifications

## AUDIO section

Effective power output during STEREO operation\*

### FRONT

(118 Hz ~ 20 kHz, 0.3 % T.H.D. at one channel driven) ..... 55W + 55 W

### SUBWOOFER

(22 Hz ~ 105 Hz, 1.5 % T.H.D. at one channel driven) ..... 102 W

Effective power output during SURROUND operation\*

### FRONT

(114 Hz ~ 20 kHz, 0.3 % T.H.D. at one channel driven) ..... 55W + 55 W

### CENTER

(118 Hz ~ 20 kHz, 0.3 % T.H.D. at one channel driven) ..... 55 W

### SURROUND

(140 Hz ~ 20 kHz, 0.3% T.H.D. at one channel driven) ..... 16 W + 16 W

### SUBWOOFER

(22 Hz ~ 105 Hz, 1.5 % T.H.D. at one channel driven) ..... 102 W

### FRONT

(1 kHz, 10 % T.H.D. at one channel driven) ..... 70 W + 70 W

### CENTER

(1 kHz, 10 % T.H.D. at one channel driven) ..... 70 W

### SURROUND

(1 kHz, 10 % T.H.D. at one channel driven) ..... 24 W + 24 W

### SUBWOOFER

(100 Hz, 10 % T.H.D. at one channel driven) ..... 110 W

\*Please note that MaxxBass processing will psycho-acoustically extend perceived low frequency response.

## Frequency response

Line (Optical 1, Optical 2, Coaxial, Analog)

..... 10 Hz ~ 100 kHz, 0 dB ~ -3.0 dB

## Tone control

BASS..... ±12 dB (at 100 Hz)

TREBLE ..... ±12 dB (at 10 kHz)

## DIGITAL AUDIO section

Sampling frequency ..... 32 kHz, 44.1 kHz, 48 kHz

Input level / impedance / wave length

Optical ..... (-15 dBm ~ -24 dBm), 700 nm ±30 nm

Coaxial ..... 0.5 Vp-p / 75 ohms

## GENERAL

Power consumption Xbox 5.1 ..... 3.5 A

Xbox 5.1 ..... 420 W

Standby Power consumption ..... less than 60W

# Texas Instruments TMS320

From Wikipedia, the free encyclopedia



Texas Instruments TMS32020.



The [ZMD U320C20](#) is a clone of the TMS320C20.

**Texas Instruments TMS320** is a blanket name for a series of [digital signal processors](#) (DSPs) from [Texas Instruments](#). It was introduced on April 8, 1983 through the TMS32010 processor, which was then the fastest DSP on the market.

The processor is available in many different variants, some with [fixed-point arithmetic](#) and some with [floating point arithmetic](#). The floating point DSP TMS320C3x, which exploits [delayed branch logic](#), has as many as three [delay slots](#).

The flexibility of this line of processors has led to it being used not merely as a co-processor for [digital signal processing](#) but also as a main CPU. Newer implementations support standard IEEE [JTAG](#) control for boundary scan and/or in-circuit debugging.

The original TMS32010 and its subsequent variants is an example of a CPU with a [modified Harvard architecture](#), which features separate address spaces for instruction and data memory but the ability to read data values from instruction memory. The TMS32010 featured a fast [multiply-and-accumulate](#) useful in both DSP applications as well as transformations used in computer graphics. The graphics controller card for the [Apollo Computer DN570 Workstation](#), released in 1985, was based on the TMS32010 and could transform 20,000 2D vectors/second. <sup>[*clarification needed*]</sup>

## Variants[\[edit\]](#)

---

The TMS320 architecture has been around for a while so a number of product variants have developed. The product codes used by Texas Instruments after the first TMS32010 processor have involved a very popular series of processor named TMS320Cabcd where *a* is the main series, *b* the generation and *cd* is some custom number for a minor sub-variant.

For this reason people working with DSPs often abbreviate a processor as "C5x" when the actual name is something like TMS320C5510, since all products obviously have the name "TMS320" and all processors with "C5" in the name are [code compatible](#) and share the same basic features. Sometimes you will even hear people talking about "C55x" and similar subgroupings, since processors in the same series and same generation are even more similar.

### Outside the main series

#### DA variants[\[edit\]](#)

DA variants (target "Digital audio")

DA610/601 is a processor with a **C67x core**. It has on chip peripherals needed to connect to audio codecs for a 5.1 or 7.1 system. This chip is used in the YAMAHA high end receivers : RX-V1400, RX-V2400, RX-V1600, RX-V2600, RX-V2500.<sup>[3]</sup>

## Software support[\[edit\]](#)

---

The TMS320 series can be programmed using [C](#), [C++](#), and/or [assembly language](#). Most work on the TMS320 processors is done using Texas Instruments proprietary [toolchain](#) and their [integrated development environment Code Composer Studio](#), which includes a mini [operating system](#) called [DSP/BIOS](#). Additionally, a department at the [Chemnitz University of Technology](#) has developed preliminary support for the TMS320C6x series in the [GNU Compiler Collection](#).<sup>[4]</sup>

In November 2007 TI released part of its toolchain as [freeware](#) for non-commercial users, offering the bare compiler, assembler, optimizer and linker under a proprietary license.<sup>[5][6]</sup> However, neither the IDE nor a debugger were included, so for debugging and JTAG access to the DSPs, users still need to purchase the complete toolchain.

In 2010 Texas Instruments contracted [CodeSourcery](#) (the assignment later transferred to [Mentor Graphics](#) as part of their acquisition) to provide deep integration and support for the C6x series in GCC, as part of their effort to port the [Linux kernel](#) to C6x. This culminated in C6x being a supported architecture in GCC release 4.7 on March 22, 2012.<sup>[7]</sup>

## See also[\[edit\]](#)

---

- [XDAIS algorithms](#)
- [CEVA, Inc.](#)
- [Qualcomm Hexagon](#)

## References[\[edit\]](#)

---

- Jump up**  http://www.ti.com/lit/ug/spru131g/spru131g.pdf
- Jump up**  this "LinuxDevices article". Archived from the original on 2013-01-28. includes more information about this platform
- Jump up**  this http://members.cox.net/alexhardware/IC\_database1.htm site includes more information

4. [Jump up](#)<sup>^</sup> Jan Parthey and Robert Baumgartl, *Porting GCC to the TMS320-C6000 DSP Architecture*, Appeared in the Proceedings of GSPx'04, Santa Clara, September 2004, [1]
5. [Jump up](#)<sup>^</sup> "TI frees its DSP toolchain". Archived from the original on 2013-01-27.
6. [Jump up](#)<sup>^</sup> Free DSP Compiler Available
7. [Jump up](#)<sup>^</sup> GCC 4.7 Release Series - Changes, New Features, and Fixes

## External links[[edit](#)]

---

- [DSP product tree](#) at Texas Instruments
- [Texas Instruments enters the DSP market](#) historical article from TI
- [C2000 low cost experimenter kits](#)
- [c6000 Discussion Forum](#) at DSPRelated.com
- [Linux-C6x](#) a top page for the recent (as of 2012) GCC and Linux ports to C6x
- [\[2\]](#) memoir by T.I. manager on creation of TMS32010 Digital Signal Processor

## XDAIS algorithms

---

From Wikipedia, the free encyclopedia

**XDAIS** or **eXpressDsp Algorithm Interoperability Standard** is a standard for algorithm development by [Texas Instruments](#) for the [TMS320](#) DSP family. The standard was first introduced in 1999 and was created to facilitate integration of DSP algorithms into systems without re-engineering cost. The XDAIS standard address the issues of algorithm resource allocation and consumption on a DSP. Algorithms that comply with the standard are tested and awarded an "eXpressDSP-compliant" mark upon successful completion of the test

The standard consists of a set of general rules and guidelines that should be applied to all algorithms. For instance, all XDAIS compliant algorithms must implement an Algorithm Interface, called IALG. For those algorithms utilizing [DMA](#), the IDMA interface must be implemented. Further, specific rules are provided for each family of TI DSP.

Problems are often caused in algorithm by hard-coding access to system resources that are used by other algorithms. DAIS prohibits the use of this type of hard-coding. Instead, DAIS requires a standard API for the application to call a particular algorithm class. This API is defined in the [xDM standard](#), also referred to as the [VISA APIs](#) (video, imaging, speech and audio).

A XDAIS developer's kit provides the standard itself, example code, and a demonstration.

Benefits of XDAIS over non-standardised approaches include:

- Significant reduction in integration time as algorithms do not trash each others resources
- Easy comparison of algorithms from multiple different sources in the same application
- Access to broad range of compliant algorithms available from multiple TI DSP Third Parties eliminates need to custom develop complex algorithms
- Algorithms work out-of-the-box with eXpressDSP Multimedia Framework Products, such as [Codec Engine \(TI\)](#)

**See also**[[edit](#)]

---

- [eXpressDsp](#)

## External links[\[edit\]](#)

---

- [XpressDSP Algorithm Standard – xDAIS Developer's Kit and xDM](#)
- [TMS320 DSP Algorithm Standard Developer's Guide](#)



# TI Gets Into the Game



*TI's Aureus(TM) DSP Brings Audiophile Sound to the Xbox(R) 5.1 Surround Sound System Built Under License by Spherex Inc.*

*Mar 9, 2004*

DALLAS (March 9, 2004) - Providing exceptionally high quality source audio for gaming and multimedia systems, Texas Instruments (TI) (NYSE:TXN) announced today that Spherex Inc. developed the Xbox 5.1 Surround Sound System based on the Aureus™ audio digital signal processor (DSP), setting a new standard for video game audio. The six-speaker Xbox 5.1 system with state-of-the-art sound processing supplied by TI's Aureus DSP provides surround sound for the entire home theater. The new multimedia audio platform by Spherex was also a finalist in TechTV's "Best of CES" awards, given for the most innovative products introduced in the 2004 Consumer Electronics Show in Las Vegas. (See [www.ti.com/pa6](http://www.ti.com/pa6).)

Priced at less than \$500, the Xbox 5.1 system incorporates all current popular surround sound formats and multiple audio inputs, allowing simultaneous support for up to five directly connected and networked audio sources. Audio inputs can include the Xbox® video game system from Microsoft, DVD and CD players, satellite TV, MP3/WMA compressed audio, PC and Internet radio. Spherex, a subsidiary of Audio Products International Corp. (API), one of the world's largest loudspeaker manufacturers, designed the Xbox 5.1 system to take advantage of technology normally used in much more expensive audio systems. TI's **TMS320DA601** audio DSP is at the heart of the system, enabling high-fidelity sound reproduction by decoding audio standards such as Dolby Digital®, Dolby® Pro Logic II and DTS®, while still having the processing headroom for advanced acoustic features such as bass extension and audio dynamics.

"We selected TI's Aureus audio DSP for the Xbox 5.1 system because of the outstanding performance, flexibility and cost effectiveness of the device," said Alex Romanov, President and CEO. "Working with TI, Spherex has been able to achieve the best speaker system available at this price point for video gaming and audio entertainment."

## **An Immersive, Realistic Sound Experience**

The Xbox 5.1 system includes a subwoofer and five satellite speakers based on API's Mirage OMNIPOLARTM technology, which utilizes room reflections to create sound as 30 percent direct to 70 percent reflected - the same ratio found in nature. As a result, the speakers disperse sound in a 360-degree pattern, giving listeners a realistic immersive experience for gaming, movies and music playback.

"Innovative products like the Xbox 5.1 system demonstrate how TI is working with customers like Spherex to bring the best available audio technology to all multimedia applications," said Mohsin Imtiaz, marketing manager of performance audio, TI. "High-performance Aureus audio DSPs make it possible for inexpensive multimedia audio platforms like the Xbox 5.1 system to provide a listening experience previously found only in high-priced equipment."

## TI Enables More Robust Multimedia Sound

TI's Aureus audio DSP solutions for home theater and gaming products enable audio manufacturers to deliver feature-rich, realistic listening experiences across product lines from high-fidelity high-end systems to feature-driven low-cost systems. The Xbox 5.1 system uses the **DA601 DSP**, a member of the Aureus family, to produce high-end audiophile sound at a cost-effective price point for multimedia solutions. The DA601 is a 225-megahertz (MHz) floating-point processor that delivers 1800 million instructions per second (MIPS) and 32/64-bit native processing. It also features an open audio framework that provides a flexible environment in which manufacturers can easily add the features that differentiate their products and bring them to market more quickly.

# Circle Surround

**Definition:** Circle Surround is a surround sound encoding/decoding format developed and marketed by [SRS Labs \(now a part of DTS\)](#).

Circle Surround approaches surround sound in a unique way. While Dolby Digital and DTS approach surround sound from a precise directional standpoint (specific sounds emanating from specific speakers), Circle Surround emphasizes sound immersion.

To accomplish this, a normal 5.1 audio source is encoded down to two channels, then re-decoded back into 5.1 channels and redistributed back to the 5 speakers (plus subwoofer) in such a way as to create a more immersive sound without losing the directionality of the original 5.1 channel source material. Also, Circle Surround can also expand two channel source material into a full 5.1 channel surround sound listening experience.

Circle Surround is now in its second generation, referred to as Circle Surround II, which expands the original Circle Surround listening environment from five to six channels. For additional details, refer to the [Official SRS Circle Surround II Page](#)

Circle Surround was developed by a company called Rocktron as a surround format specifically designed for music. SRS purchased it from Rocktron and after years of R & D, they refined Circle Surround into a flexible world-class encode/decode solution that rivals discrete digital formats. The CS encoder is a 6.1 encode (L, C, R, LS, RS, CS, LFE)!

CSII Decoding has a Mono mode (for mono content), Music Mode (for stereo content) and Cinema mode (for surround encoded content).

CS encoded material can be decoded using PL, PLII, Neo 6, Logic 7, CS, or CSII, up to the limitations of the decoder being used.

## WHAT CSII DECODING DO FOR YOU

Mono to 6.1

Stereo to 6.1

Dolby Surround decoding to 6.1

Full bandwidth in each channel

SRS Dialog Clarity (option to improve dialog intelligibility)

SRS TruBass (option to greatly enhance bass response)

Improved steering (over older surround formats)

Improved channel separation (over older surround formats)

## TV IN CIRCLE SURROUND

ESPN recently chose Circle Surround encoding for the NFL, NHL, NBA and X-Games. Watch the next NFL game on ESPN and you'll see the "Presented in Circle Surround" logo in the upper left hand corner.

Currently, NBC is using CS with "Frasier", and CBS is doing so with "Becker". However, neither of these have logos yet (in progress). If you don't have a CS decoder, you may use PL or PLII if you'd like, but CSII provides optimum results.

Surround Expo 2002 just finished last week and response to CS was incredible.

Circle Surround allows delivery of modern day, 5.1 or 6.1 multichannel content in as little as 44kbps. We are exploring the tools necessary to allow the creation of CS encoded content for streaming 6.1 over the Internet. This would be great for things like the movie trailers at Apple's web site (hint hint...if anyone knows any decision makers at Apple that could get them to develop a QT encoding plug in for CS...I'll get them the code)

CSII has no relationship to AC3 or DTS. It can decode Dolby Surround encoded content, which is different than either of these two formats you mentioned.

Both of those are discrete and proprietary methods of decoding that serve one specific purpose. While most of us likely use AC3 or DTS for playback of DVDs, these formats are relatively inflexible in how they can be delivered. The bitstreams are too large to easily transmit over existing broadcast infrastructures, or to stream over the Internet.

For example, if you started with basic two channel stereo source material and encoded it in AC3 as stereo and assigned the channels (appropriately) to the Left and Right speakers, the only thing any AC3 decoder can do with it is play it in stereo, through the Left and Right speaker. What you encode is what you get (WYIWIYG?).

If you play an analog or digital PCM stereo signal back through a Circle Surround II decoder, you'll get a very compelling 6.1 surround from it. If it is music, and the CSII decoder is in music mode, depending on the source material, you will most likely have a strong vocal soundstage up front (through the Left, Center and Right speakers), with emphasis on instrumentation in the rears. There is still solid separation among the speakers in this mode.

It's like PLII with better, dual-band steering and separation, an improved encoding method, dialog clarity and TruBass (and an amazing ability to hold up under serious bandwidth limitations and/or compression).

## Floating point

From Wikipedia, the free encyclopedia

An early electromechanical programmable computer, the [Z3](#), included floating-point arithmetic (replica on display at [Deutsches Museum](#) in [Munich](#)).

$$1.2345 = \underbrace{12345}_{\text{mantissa}} \times 10^{\text{exponent } -4}$$



A diagram showing a representation of a decimal floating-point number using a [mantissa](#) and an [exponent](#).

In [computing](#), **floating point** describes a method of representing an approximation of a [real number](#) in a way that can support a wide range of values. The numbers are, in general, represented approximately to a fixed number of [significant digits](#) (the [significand](#)) and scaled using an [exponent](#). The base for the scaling is normally 2, 10 or 16. The typical number that can be represented exactly is of the form:

$$\text{Significant digits} \times \text{base}^{\text{exponent}}$$

The idea of floating-point representation over intrinsically [integer fixed-point](#) numbers, which consist purely of [significand](#), is that expanding it with the exponent component achieves greater range. For instance, to represent large values, e.g. distances between galaxies, there is no need to keep all 39 decimal places down to [femtometre](#)-resolution (employed in particle physics). Assuming that the best resolution is in [light years](#), only the 9 most significant decimal digits matter, whereas the remaining 30 digits carry pure noise, and thus can be safely dropped. This represents a savings of

100 [bits of computer data storage](#). Instead of these 100 bits, much fewer are used to represent the scale (the exponent), e.g. 8 bits or 2 decimal digits. Given that one number can encode both astronomic and subatomic distances with the same nine digits of accuracy, but because a 9-digit number is 100 times less accurate than the 11 digits reserved for scale, this is considered a [trade-off](#) exchanging range for [precision](#). The example of using scaling to extend the dynamic range reveals another contrast with fixed-point numbers: Floating-point values are not uniformly spaced. Small values, close to zero, can be represented with much higher resolution (e.g. one femtometre) than large ones because a greater scale (e.g. light years) must be selected for encoding significantly larger values.<sup>[u]</sup> That is, floating-point numbers cannot represent point coordinates with atomic accuracy at galactic distances, only close to the origin.

The term *floating point* refers to the fact that a number's [radix point](#) (decimal point, or, more commonly in computers, binary point) can "float"; that is, it can be placed anywhere relative to the significant digits of the number. This position is indicated as the exponent component in the internal representation, and floating point can thus be thought of as a computer realization of [scientific notation](#).

Over the years, a variety of floating-point representations have been used in computers. However, since the 1990s, the most commonly encountered representation is that defined by the [IEEE 754](#) Standard.

The speed of floating-point operations, commonly referred to in performance measurements as [FLOPS](#), is an important characteristic of a computer system, especially in [software](#) that performs large-scale mathematical calculations.

## Overview[[edit](#)]

---

A number representation (called a [numeral system](#) in mathematics) specifies some way of storing a number that may be encoded as a string of digits. The arithmetic is defined as a set of actions on the representation that simulate classical arithmetic operations.

There are several mechanisms by which strings of digits can represent numbers. In common mathematical notation, the digit string can be of any length, and the location of the [radix point](#) is indicated by placing an explicit ["point" character](#) (dot or comma) there. If the radix point is not specified then it is implicitly assumed to lie at the right (least significant) end of the string (that is, the number is an [integer](#)). In [fixed-point](#) systems, some specific assumption is made about where the radix point is located in the string. For example, the convention could be that the string consists of 8 decimal digits with the decimal point in the middle, so that "00012345" has a value of 1.2345.

In [scientific notation](#), the given number is scaled by a [power of 10](#) so that it lies within a certain range—typically between 1 and 10, with the radix point appearing immediately after the first digit. The scaling factor, as a power of ten, is then indicated separately at the end of the number. For example, the revolution period of [Jupiter's moon Io](#) is 152853.5047 seconds, a value that would be represented in standard-form scientific notation as  $1.528535047 \times 10^5$  seconds.

Floating-point representation is similar in concept to scientific notation. Logically, a floating-point number consists of:

- A signed (meaning positive or negative) digit string of a given length in a given [base](#) (or [radix](#)). This digit string is referred to as the [significand](#), [coefficient](#) or, less often, the mantissa (see below). The length of the significand determines the *precision* to which numbers can be represented. The radix point position is assumed to always be somewhere within the significand—often just after or just before the most significant digit, or to the right of the rightmost (least significant) digit. This article will generally follow the convention that the radix point is just after the most significant (leftmost) digit.
- A signed integer [exponent](#), also referred to as the characteristic or scale, which modifies the magnitude of the number.

To derive the value of the floating-point number, one must multiply the *significand* by the *base* raised to the power of the *exponent*, equivalent to shifting the radix point from its implied position by a number of places equal to the value of the exponent—to the right if the exponent is positive or to the left if the exponent is negative.

Using base-10 (the familiar [decimal](#) notation) as an example, the number 152853.5047, which has ten decimal digits of precision, is represented as the significand 1.528535047 together with an exponent of 5 (if the implied position of the radix point is after the first most significant digit, here 1). To determine the actual value, a decimal point is placed after the first digit of the significand and the result is multiplied by  $10^5$  to give  $1.528535047 \times 10^5$ , or 152853.5047. In storing such a

number, the base (10) need not be stored, since it will be the same for the entire range of supported numbers, and can thus be inferred.

Symbolically, this final value is

$$s \times b^e$$

where  $s$  is the value of the significand (after taking into account the implied radix point),  $b$  is the base, and  $e$  is the exponent.

Equivalently:

$$\frac{s}{b^{p-1}} \times b^e$$

where  $s$  here means the integer value of the entire significand, ignoring any implied decimal point, and  $p$  is the precision—the number of digits in the significand.

Historically, several number bases have been used for representing floating-point numbers, with base 2 ([binary](#)) being the most common, followed by base 10 (decimal), and other less common varieties, such as base 16 ([hexadecimal notation](#)), as well as some exotic ones like 3 (see [Setun](#)).

Floating-point numbers are [rational numbers](#) because they can be represented as one integer divided by another. For example  $1.45 \times 10^3$  is  $(145/100) \times 1000$  or  $145000/100$ . The base however determines the fractions that can be represented. For instance,  $1/5$  cannot be represented exactly as a floating-point number using a binary base but can be represented exactly using a decimal base ( $0.2$ , or  $2 \times 10^{-1}$ ). However  $1/3$  cannot be represented exactly by either binary ( $0.010101\dots$ ) nor decimal ( $0.333\dots$ ), but in [base 3](#) it is trivial ( $0.1$  or  $1 \times 3^{-1}$ ). The occasions on which infinite expansions occur depend on the base and its [prime factors](#), as described in the article on [Positional Notation](#).

The way in which the significand, exponent and sign bits are internally stored on a computer is implementation-dependent. The common IEEE formats are described in detail later and elsewhere, but as an example, in the binary single-precision (32-bit) floating-point representation  $p=24$  and so the significand is a string of 24 [bits](#). For instance, the number  $\pi$ 's first 33 bits are 11001001 00001111 11011010 10100010 0. Given that the 24th bit is zero, rounding to 24 bits in binary mode means attributing the 24th bit the value of the 25th which yields 11001001 00001111 11011011. When this is stored using the IEEE 754 encoding, this becomes the significand  $s$  with  $e = 1$  (where  $s$  is assumed to have a binary point to the right of the first bit) after a left-adjustment (or *normalization*) during which leading or trailing zeros are truncated should there be any. Note that they do not matter anyway. Then since the first bit of a non-zero binary significand is always 1 it need not be stored, giving an extra bit of precision. To calculate  $\pi$  the formula is

$$\begin{aligned} & \left( 1 + \sum_{n=1}^{p-1} \text{bit}_n \times 2^{-n} \right) \times 2^e \\ &= \left( 1 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-4} + 1 \times 2^{-7} + \dots + 1 \times 2^{-23} \right) \times 2^1 \\ &= 1.5707964 \times 2 \end{aligned}$$

where  $n$  is the normalized significand's  $n$ -th bit from the left. Normalization, which is reversed when 1 is being added above, can be thought of as a form of compression; it allows a binary significand to be compressed into a field one bit shorter than the maximum precision, at the expense of extra processing.

The word "mantissa" is often used as a synonym for significand. Use of mantissa in place of significand or coefficient is discouraged, as the mantissa is traditionally defined as the fractional part of a logarithm, while the *characteristic* is the integer part. This terminology comes from the manner in which [logarithm](#) tables were used before computers became commonplace. Log tables were actually tables of mantissas.

## MaxxBass

an algorithm to create the sense of the missing fundamental by synthesizing higher harmonics.<sup>[13]</sup> Waves Audio released the MaxxBass [plug-in](#) to allow computer users to apply the synthesized harmonics to their audio files. Later, Waves Audio produced small [subwoofers](#) that relied on the missing fundamental concept to give the illusion of low bass.<sup>[14]</sup> Both products processed certain overtones selectively to help small loudspeakers, ones which could not reproduce low-frequency components, to sound as if they were capable of low bass. Both products included a [high-pass filter](#) which greatly attenuated all the low frequency tones that were expected to be beyond the capabilities of the target sound system.<sup>[15]</sup>

## VelociTI Architecture

**...extensive parallelism and pipelining-which is scheduled by the development tools.**

**... provides eight execution units, including two multipliers and six arithmetic logic units (ALUs). These units operate in parallel...**

**... VelociTI's advanced features include instruction packing, conditional branching, and pre-fetched branching, all of which overcome problems that were associated with previous VLIW implementations. The architecture is highly deterministic, with few restrictions on how or when instructions are fetched, executed, or stored.**

## AUDIO section

Effective power output during STEREO operation\*

**FRONT**  
(118 Hz ~ 20 kHz, 0.3 % T.H.D. at one channel driven) ..... 55W + 55 W  
**SUBWOOFER**  
(22 Hz ~ 105 Hz, 1.5 % T.H.D. at one channel driven) ..... 102 W

Effective power output during SURROUND operation\*

**FRONT**  
(114 Hz ~ 20 kHz, 0.3 % T.H.D. at one channel driven) ..... 55W + 55 W  
**CENTER**  
(118 Hz ~ 20 kHz, 0.3 % T.H.D. at one channel driven) ..... 55 W  
**SURROUND**  
(140 Hz ~ 20 kHz, 0.3% T.H.D. at one channel driven) ..... 16 W + 16 W  
**SUBWOOFER**  
(22 Hz ~ 105 Hz, 1.5 % T.H.D. at one channel driven) ..... 102 W

**FRONT**  
(1 kHz, 10 % T.H.D. at one channel driven) ..... 70 W + 70 W  
**CENTER**  
(1 kHz, 10 % T.H.D. at one channel driven) ..... 70 W  
**SURROUND**  
(1 kHz, 10 % T.H.D. at one channel driven) ..... 24 W + 24 W  
**SUBWOOFER**  
(100 Hz, 10 % T.H.D. at one channel driven) ..... 110 W

\*Please note that MaxxBass processing will psycho-acoustically extend perceived low frequency response.

## Frequency response

Line (Optical 1, Optical 2, Coaxial, Analog)  
..... 10 Hz ~ 100 kHz, 0 dB ~ -3.0 dB

## Tone control

BASS ..... ±12 dB (at 100 Hz)  
TREBLE ..... ±12 dB (at 10 kHz)

## DIGITAL AUDIO section

Sampling frequency ..... 32 kHz, 44.1 kHz, 48 kHz  
Input level / impedance / wave length  
Optical ..... (-15 dBm ~ -24 dBm), 700 nm ±30 nm  
Coaxial ..... 0.5 Vp-p / 75 ohms

## GENERAL

Power consumption Xbox 5.1 ..... 3.5 A  
Xbox 5.1 ..... 420 W  
Standby Power consumption ..... less than 60W

## DIMENSIONS

Satellites:  
HxLxW: 5-1/2" x 6-1/2" x 3-1/4"  
14 x 16.5 x 8.3cm  
Weight: 2-1/2 lbs.  
1.1 kg  
  
Subwoofer:  
HxLxW: 16" x 9-1/4" x 16"  
40.6 x 23.5 x 40.6cm  
Weight: 28 lbs.  
12.7 kg  
  
Over Carton:  
LxWxH: 21-1/4" x 20-3/4" x 19-1/4"  
54 x 52.7 x 48.9cm  
Weight: 60 lbs.  
27.2 kg

Specifications Subject to Change Without Notice.