

Vesta Product History

STD88

STD-bus processor card. Intel 8088 processor at 4MHz. 32KB EPROM, 32KB SRAM. Bit-bang serial interface (limited to 4800 baud). Peripherals developed later included bubble memory card. Programming languages were figForth and an integer BASIC interpreter.

SBC88

Standalone SBC. Intel 8088 processor at 4MHz. 32KB EPROM, 8-24KB SRAM, 8KB EPROM programmer socket. Bit-bang serial interface (limited to 4800 baud). Eight-channel 8- or 10-bit ADC. Standard DIP sockets for I/O connections. A later revision of this board (the SBC88A) used standard 0.1" headers for connections. There were a couple of different power supplies developed for this board. Programming languages were figForth and an integer BASIC interpreter. Later, figForth was replaced by Forth-83, and support for C was added. Other variants offered used NEC V20 processors (CMOS, capable of 8MHz). The higher speed allowed serial I/O at 9600 baud.

OEM188

Standalone SBC. 8"x8" PCB. Intel 80188 processor at 8MHz. 64KB EPROM, 8-256KB SRAM (256KB using 32KB SRAM devices in 2-high stacking sockets). Floppy controller, 2-channel UART, and interrupt controller chip on board. This board was used as the basis for the article series in Radio-Electronics Magazine that defined a robot (the RERB). Programming languages were figForth and an integer BASIC interpreter. Later, figForth was replaced by Forth-83, and support for C was added. For a time, a multitasking BASIC with floating-point support was offered. This was the last board laid out using tape and Mylar at Vesta.

Tiny188

Standalone SBC. Intel 80188 processor at 8MHz. Onboard 2-channel UART (one channel of which could be configured as either RS232 or RS422) and peripheral I/O chip. Programming languages were figForth and an integer BASIC interpreter. Later, figForth was replaced by Forth-83, and support for C was added. A later board variant (Tiny188A) added a real-time clock/calendar chip onboard, and increased the amount of memory that could be used. Several peripheral boards were developed.

Micro88

Standalone SBC. Intel 8088 processor at 4MHz. Onboard 2-channel UART (one channel of which could be configured as either RS232 or RS422) and peripheral I/O chip. Programming languages were figForth and an integer BASIC interpreter. Later, figForth was replaced by Forth-83, and support for C was added. A later board

variant (Micro88A) changed the connectors, and increased the amount of memory that could be used. Several peripheral boards were developed. NEC V20 processors were also an option on this board

SBC68K

Developed when we were working on a multi-processor exercise machine because the Tiny188 didn't have enough horsepower. Used an 8MHz 68000 processor. Programmed in Forth83 (ported from an Amiga PD Forth).

SBC196

Standalone SBC. Used an Intel 80196 processor (KB and KC variants). Programmed in Forth83i196 (a Forth83 variant specific to the board). The Forth was developed on the Amiga, which was used as a development platform at Vesta for a few years. Two serial channels (one bit-bang). Available in 8MHz, 10MHz, 12MHz, and 16MHz (KC) versions, each of which required a different language EPROM. C was added as a development language later. Memory map depended on a PAL, with different PALs required for C and Forth. Several peripheral boards were developed, including memory and I/O expansion boards.

SBC332

Standalone SBC modeled after the Motorola BCC (Business Card Computer) demo board for the 68332 processor. Ours was the same width, but longer, and used the same connector configuration (both pinout and physical layout) for bus expansion. Up to 1MB EPROM, 1MB SRAM. Motorola 68332A processor at up to 16MHz (configurable under program control). Programming was done in VFSE (Vesta Forth, Standard Edition) or C.

SBC2000

Around 1995, we decided to design a new product line which would use common connectors, consistent pinout conventions (for example, pin 1 on the power connector is always ground), and common peripherals for which support would be built into the SBCs. We had been using a lot of peripherals that used synchronous serial interfaces, because they allowed us to use fewer processor I/O pins for a given amount of I/O. The three most common synchronous serial interfaces are SPI (Serial Peripheral Interface, defined by Motorola), IIC (or I²C, the Inter-Integrated Circuit bus, defined by Philips), and MicroWire (defined by National Semiconductor). We defined VAST (the Vesta Addressable Synchronous Transfer bus) to support both SPI and IIC communications. Four I/O lines provided at the connector gave us 16 different "addresses" on the bus. Address 15 was reserved for IIC, and addresses 0-14 were for SPI. MicroWire was very similar to SPI, but not as popular, so we didn't support it directly. A number of peripherals (digital I/O, analog input and output, relay boards, display boards, etc.) were developed. Each board took from ¼ to 2 VAST addresses.

SBC2000-332

This was one of the first two products in the SBC2000 line. The SBC2000-332 used a Motorola 68332G operating at up to 20MHz (later 25MHz), and offered up to 1MB each of EPROM, Flash, and SRAM. 4KB or 8KB of EEPROM and a clock/calendar chip were also on the board. Standard peripherals included two serial ports (one of which could be configured as RS232 or RS422), LCD, keypad, and VAST. A bus interface almost compatible with the PC104 standard was also provided, but only two peripherals were ever developed (an 8-channel 12-bit ADC, and a 4-port serial I/O board). Programming language was Vesta Basic (later Vesta MultiTasking Basic, and later VMTB). Support for C was added later – we supported SDSI's CrossCode C toolset, and our customers also used Intermetrics and GNU C.

SBC2000-188

This was the other of the first two products in the SBC2000 line. The SBC2000-188 used an Intel 80188EB processor clocked with a 40MHz crystal (20MHz operation), and offered up to 1MB of total memory space (if EPROM + SRAM totalled less than 1MB, the space in between was mapped as bus expansion. 4KB or 8KB of EEPROM and a clock/calendar chip were also on the board. Standard peripherals included two serial ports (one of which could be configured as RS232 or RS422), LCD, keypad, and VAST. A bus interface almost compatible with the PC104 standard was also provided, but only two peripherals were ever developed (an 8-channel 12-bit ADC, and a 4-port serial I/O board). Programming language was Vesta Basic (later Vesta MultiTasking Basic, and later VMTB). Support for C was added later. We supported Borland C using the Paradigm ROMable toolset. Paradigm would also support Microsoft C, and later converted to their own C/C++ compiler.

SBC2000-074

This board was developed because many applications didn't need the power of the SBC2000-332 or SBC2000-188. The system used a Microchip PIC16C74 (later, PIC16F74B) running at 20MHz, and could only be programmed in Vesta SingleTasking Basic (later, VSTB). Standard peripherals included 2 serial ports, 8-channel 8-bit ADC, LCD, keypad, and VAST. Program memory was 8KB to 32KB of EEPROM, and a user application had access to about 50 bytes of variable space.

SBC2000-062

This board was developed at the same time as the SBC2000-074, and was meant for applications that had truly small processing requirements. It used a Microchip 16C62 processor running at 5MHz, and could only be programmed in Vesta SingleTasking Basic (later, VSTB). Standard peripherals included 1 serial port (bit-bang, 9600 baud), LCD, and keypad. It offered less RAM than the SBC2000-074. A variant using the PIC16C72 processor provided an 8-channel, 8-bit ADC.

MC2000-332

To allow for quicker development of custom applications, a variant of the SBC2000-332 was developed. This was a smaller board, and did not have the separate LCD, VAST, etc. connectors, nor were the address and data busses made available. It was designed to plug into an ASB (Application-Specific Board) which provided the peripherals the customer needed. The idea was that for one-off or prototype

systems, an SBC2000-332 with VAST peripherals would be used, and the MC2000-332 would be used with an ASB that incorporated the peripherals when the system went into production. The MC2000-332 also incorporated an 8-channel, 12-bit ADC on the board. The board is about the size of a credit card, thus the name (MC stands for MasterCard).

MC2000-074

This was developed to have the same relationship to the SBC2000-074 that the MC2000-332 bore to the SBC2000-332. The board was about 1" square.

MC2000-077

This was developed because the founder of the company got into R/C aircraft as a hobby. He developed several autopilots – one based on the MC2000-332, and one based on the MC2000-074. The MC2000-074 didn't offer as much variable space as he wanted, so the MC2000-077 was developed. This provides about 100 bytes of global variables, and has other enhancements to VSTB. The -077 was used to develop the AP40 and AP50.

Languages

Tiny BASIC

This was Vesta's original programming language offering. It was developed from a Tiny BASIC interpreter that had been published in Dr. Dobb's Journal. It supported 52 variables (A-Z, and A0-Z0). Originally, it ran on top of a CPM-86 BIOS. When Forth83 was developed, it used an IBM PC BIOS, so BASIC was ported to run on that for consistency among the products. It went through continual development and enhancement for a number of years. The OEM188 and Tiny188 variants of the language included disk access routines.

figForth

This was Vesta's second programming language offering. figForth was developed to make public-domain implementations of the Forth programming language widely available. It ran *much* faster than the Tiny BASIC, and was a more powerful language. It also ran on top of a CPM-86 BIOS.

Forth83 (all variants)

figForth was effectively a 1978 standard for Forth. The 1983 standard became rather popular, so we developed a port to our 80x88-based boards. We started with the most popular public-domain Forth83, written by Laxen and Perry. Because this was an MS-DOS Forth, it was expecting a PC BIOS at the lowest level. Rather than port it to the CPM-86 BIOS we were using at the time, we developed PC-type BIOS implementations for all of our boards.

This Forth supported some advanced capabilities, such as multitasking and redirectible I/O. We added some features of our own, such as the capability of booting from precompiled applications, which reduced the boot time for a system. When we rewrote the Forth to use direct-threading instead of indirect-threading (a significant performance boost), we changed the name to Forth83+.

Forth83i96 was the variant of Forth83 that was developed for the SBC196. This version was written from scratch, and cross-compiled from an Amiga.

Another variant ran on the SBC68K. This was developed from a Forth that ran on the Amiga, and had been derived from Laxen & Perry Forth originally.

VFSE

This is the Forth that was developed for the SBC332. We wrote it from scratch, and based it on the ANSI standard for Forth, which was under development at the time. The core of the system was written in assembly language, and the Forth compiler used subroutine-threading to increase execution speed. It supported a number of advanced features, including the ability to save relocated images of the system plus any compiled application code in RAM, such that new EPROMs could be made that would boot an application instantly.

VMTB

This is the current name for the Basic that runs on the -332 and -188 boards in the SBC2000 product line. It is compiled to p-codes, which makes it faster than a pure interpreter, but slower than a machine-code compiler. Multitasking is a basic feature of the language. Up to eight tasks are allowed in a system, with task 0 being reserved for the debug monitor program. The language supports 16-bit and 32-bit integers, 32-bit floats, and strings up to 253 characters long. Subroutines and functions have argument lists and local variables, recursion is allowed, I/O redirection is possible, and event and exception-handling is built in. We have an IDE (Integrated Development Environment) for the language.

VSTB

This is the current name for the Basic that runs on the -062, -074, and -077 boards in the SBC2000 product line. The language has different capabilities, depending on which board you're compiling for, but the basic features are the same. Control structures, recursion, and statement format is designed to have maximum compatibility with VMTB. String variables are not supported, but string constants are. Event and exception handling is built in, but fewer events are supported because of board limitations. Various built-in routines are provided as part of the language because they could not be done well programmed in VSTB.

On the -062, bit, byte, and 16-bit integers are supported.

The -074 adds support for 32-bit floats.

The -077 adds support for 32-bit integers and more events.

C/Assembly

All current and past boards can be programmed in C or assembly, and some can be programmed in C++. Vesta does not support the use of C on our PIC-based boards, though.

Different toolsets are used for different processors. This is the development option to use if the absolute best performance is required for an application – VMTB provides simpler programming compared to C, at the cost of lower performance.

On the PIC-based boards, C is not a supported option, but a customer *could* use it. Because the application program is stored in EEPROM, VSTB can also provide more capability than a C application would be able to put into the same processor. Again, this is at the cost of performance. Also, all of the VSTB capabilities are present whether or not an application needs them all; a C program would only need to implement the capabilities it needed.