

## Vesta Multi-tasking Basic

for SBC2000-1888, SBC2000-332, MC2000-332 & Dev-332

### Features

- Multitasking among 7 tasks with task handling control and interrupt control
- Compiled for fast execution
- Easy to use Integrated Development Environment
- Flash upgradeable code
- Debug on the SBC with single step, animate and trace variables
- I/O can be redirected with PIPE (to serial ports, LCD, and custom routines)
- Event handling (errors, interrupts, timers, and serial ports)
- Specialized real time control extensions
- Integer, float, long, byte, string and array variables
- Floating point and transcendental math
- Portability among SBC2000 engines
- Extensible using C

### Embedded Application Development

Vesta Technology's innovative Vesta Multi-tasking Basic (VMTB) is specifically designed for embedded system development. It combines the speedy execution of a compiled language with familiar BASIC constructs, plus some special features for embedded systems design, to give you complete control of every aspect of your SBC2000 engine.

Vesta Basic helps you write cleaner code faster by providing extensive, easy to use keywords and object oriented structures that are versatile enough for any application.

Vesta Technology gives you the power, flexibility and ease of use that make application development fun again.

### Structured Programming

Modeled on the latest, third-generation BASICs, Vesta Multi-tasking Basic is procedure oriented to promote structured programming, enhance productivity and minimize maintenance. Subroutines and functions are called by name. Fully descriptive variable and procedure names are supported. Global, local, static and kernel variables all promote information containment.

Vesta Basic is optimized to do more with less code. The keywords are powerful, the constructs are terse, and the compiler is intelligent. You'll be amazed at how much you can squeeze into your application.

### Speed

Vesta Basic source code is compiled into tokens that are executed by the Runtime Engine on the Vesta SBC2000 single-board computer to give you the speedy execution needed for real-time embedded control. On the SBC2000-332 and MC2000-332 engines, integer calculations run at about 38,500 lines/second, and execute on the SBC2000-188 at about 15,500 lines/second.

### Task and Interrupt Control

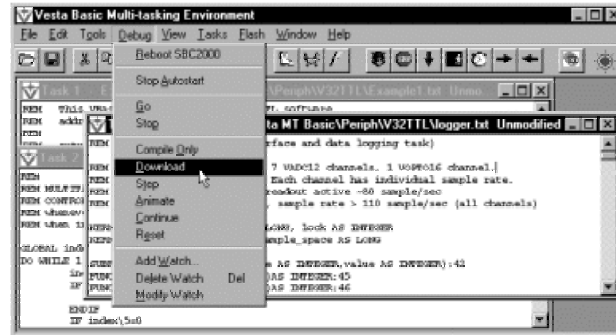
Vesta Multi-tasking Basic provides 8 software timers, ON INT0 and ON ERROR interrupts, and an ON COM interrupt for each of the two default serial ports (additional serial ports can be added if needed).

Task switching and interrupt handling can be suppressed and restarted. Tasks can call, stop, and restart each other or themselves. Tasks can give themselves more time or relinquish the remainder of their time to the next task.

# Vesta Multi-tasking Basic - page 2

## The Integrated Development Environment

The IDE greatly simplifies the usual tasks of the embedded systems programmer by wrapping them in the familiar Windows environment, and reducing formerly tedious operations to the click of a mouse button. A single click of the debug button compiles, links, and downloads your code and invokes the debugger.



## Powerful Animated DeBug

Vesta Multi-tasking Basic allows you to debug your code as it will run in the actual destination environment. You can download your program to the SBC and step through each, line watching variables as they change.

You can observe and manipulate variables and can even send control characters to the SBC. Errors are reported in real time, giving you power that embedded developers rarely experience.

There is simply no substitute for knowing exactly how your code will execute in the environment for which it is written.

## Task Status Window



The IDE tracks where each task is located on the SBC, and allows you to erase flash to regain control from a runaway task.

A Task Status window is available to view the status of each task. The status indicator actually polls the SBC2000 to read the status of the tasks. Thus, tasks that have been stopped by other tasks will be correctly reported.

## Link Activity Indicator

The heart/sun activity indicator in the upper right corner indicates polling activity between the Vesta Basic IDE and the SBC.

## Support of VAST Peripherals

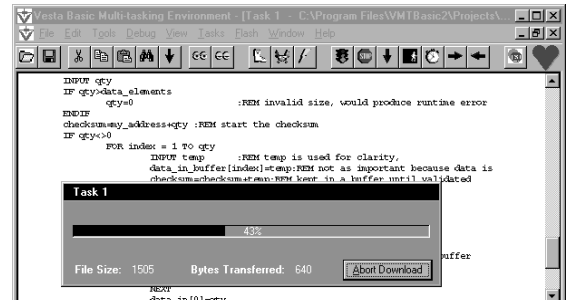
Vesta SBC2000 embedded controllers are designed to work with our complete line of VAST (Vesta Asynchronous Serial Transfer) and Bus peripherals.

Each Vesta peripheral is supported by its own external procedures that are added to the language as needed. These procedures have been carefully designed to keep the language straightforward and easy to read, yet still allow the programmer to access all of the potential of the hardware.

AIN\_CONFIG selects the A/D peripheral board and mode of use (single ended vs. differential, unipolar vs. bipolar)

AIN(channel) returns the result.

This saves dozens of lines of code, and more importantly, your time.



# Vesta Multi-tasking Basic - page 3

## Flash Upgrades of Microcontroller and Application

The Vesta Basic runtime engine is normally stored in flash, and can be upgraded by the IDE. Tasks can be stored in flash for automatic execution on power-up and easy replacement. The runtime engine and tasks can also be put into EPROM.

During development any task stored in boot ROM or flash may be overridden by downloading a new version of that task into RAM where it may be executed under the control of the IDE. After development is complete the task may be moved to onboard flash or Vesta Basic can produce a Program Image so that a new boot ROM can be burned externally.



## Easy SPI and IIC Transfer

Communication with Vesta's many VAST (Vesta Asynchronous Serial Transfer) peripherals, or any SPI or IIC device, is made easy with specialized commands that allow you to address a multitude of devices with just a few simple routines.

## Powerful Substring Pointers

Substring pointers allow complex manipulation of strings with elegant, reusable code

## Power Management

Vesta's SBC2000 products are well suited for low power devices such as those powered by battery or solar cells. Vesta Basic's power management commands allow you to adjust power consumption for varying conditions.

## Extensible Language

Vesta Basic is extensible. Named external procedures may be added to Vesta Basic by coding the driver in C (or assembler called from C) using the format established in our C library, and linking the extension into the language. This means you can write drivers for your specialized hardware and make those devices as fast and easy to use as ours.

## Extensive Code Examples

The Vesta Basic CD contains many examples of programming techniques for such things as sharing data between tasks, accessing various types of devices, and controlling any of our VAST or Bus peripheral boards.

## Minimum System Requirements

Vesta Basic requires a PC with a '386 or higher microprocessor, Windows 95/97/NT, one available COM port, 8 megabytes of RAM and 10 megabytes of available hard disk space.

## Language

Vesta Multi-tasking Basic provides extensive, easy to use keywords and control structures that are versatile enough for any application and object oriented to allow easy creation of extremely powerful programs. Here is a partial vocabulary listing of the Vesta Multi-tasking Basic keywords and some built-in procedures.

# Vesta Multi-tasking Basic - page 4

## Development Kits

Vesta Development Kits give you everything you need to get started developing your project and at a fraction of the cost.

### Development kits

Vesta Development kits give you everything you need to get started developing your project, and at a fraction of the cost.

Description	DK3	DK5	DK11
SBC2000	-188	-332	
MC2000 & Dev platform			-332
Vesta MT Basic CD	?	?	?
Hard copy manuals	?	?	?
Boot ROM	?	?	?
Power Supply	?	?	?
Beeper	?	?	On Dev
Keypad		?	?
RS-232 Dev Cable	?	?	?
4x20 LCD & LCD Cable	?	?	?

<b>Compilation Control</b> INCLUDE REM	<b>I/O Space</b> IOPEEK IOPOKE	<b>LCD Functions</b> LCD_COMMAND LCD_DISPLAY	<b>Power Management</b> NAP SLEEP HIBERNATE	<b>Array Manipulation</b> SIZE_OF ADDR_OF ROWS_OF COLS_OF
<b>Flash Memory</b> FLASH_DPEEK FLASH_DPOKE FLASH_ERASE FLASH_AVL	<b>Assignment</b> LET DATA, READ, RESTORE INPUT DIM	<b>Interrupt Control</b> ON TIMERx ON ERROR ON INTx ON COMx	<b>Memory</b> RAM_AVL PEEK POKE DPEEK DPOKE	<b>Real-Time Clock</b> TIMEDATE_PEEK TIMEDATE_POKE RTC_PEEK RTC_POKE
<b>Onboard Peripherals</b> KEYPAD BEEP PWM COUNT PRINT PIPE	<b>Serial Port Control</b> COMM MODE PTT KEY_PENDING INKEY GET_TIMEOUT	<b>Task Control</b> START STOP RUN SUSPEND SET_TIMESLICES REBOOT	<b>Declaration</b> GLOBAL KERNEL LOCAL STATIC CONSTANT DECLARE	<b>Math Operators</b> MIN MAX SIN COS ARCTAN ABS LOG
<b>Control Loops</b> IF, [ELSE], ENDIF SELECT, CASE, [ELSE], ENDSELECT DO WHILE, [EXIT], LOOP DO, [EXIT], LOOP UNTIL FOR [STEP], [EXIT], NEXT EXIT	<b>String Manipulation</b> HEX DEC ASC CHR VAL FVAL STR LEN FSTR	<b>Functions and Subroutines</b> [CALL], SUBROUTINE, [RETURN] FUNCTION, [RETURN] CRITICAL VITAL END	<b>Off-Board Memory</b> MEM_PEEK MEM_POKE MEM_DPEEK MEM_DPOKE EEPROM EEPROM_SIZE EEPROM_PEEK EEPROM_POKE	<b>Asynchronous Serial Communication</b> VAST_CLOCK() VAST_OPEN() VAST_CLOSE() VAST_SPI_XFR() VAST_IIC_START() VAST_IIC_STOP() VAST_IIC_SEND() VAST_IIC_READ()