MINT 709, Capstone Project

Market Survey, Selection, and Evaluation, of a Single Board Computer for an Underwater Sensor Net Node

Written by, Stephen W. Hosier
May, 2006

1st Reader: Professor M. MacGregor

2nd Reader: Professor J. Harms

Abstract

In 2001 glass sponge reefs were discovered in the Straight of Georgia off the coast of British Columbia. Very little was known about why theses sponges grew where they did so a sensor net was needed to gather long-term data about the environmental conditions in areas where these glass sponge reefs were found. An underwater network called VENUS that became operational 8 February 2006 was located in close proximity to the area so it was possible to deploy a sensor net that connected to the VENUS system. This paper is concerned with a market survey, selection and evaluation of a single board computer (SBC) for use as a node for the glass sponge reef sensor net.

Table of Contents

1. Introduction

1.1. Glass Sponge Reefs

Glass sponge reefs were discovered in 2001 off the coast of British Columbia. The following article by Gitai Yahel [26] explains what glass sponge reefs are and how they were discovered.

The Strait of Georgia Glass Sponge Reefs
by Gitai Yahel (Yahel@UVic.ca)

During a multibeam bathymetric survey in 2001, Kim Conway along with a group of geologists from the Pacific Geosciences Center (PGC), discovered two glass sponge reefs in the Strait of Georgia. The presence of these spectacularly dense populations of giant sponge in the turbid water off the Fraser River pro-delta located just a few km from Vancouver came as a complete surprise. The known glass sponge reefs, discovered in the Hecate Strait in the early 90s, occur in a relatively low sedimentation setting. The Fraser Ridge reef consists of several interconnected mounds that are up to 14 m high and form an ~1km long belt. The reef lies on the crest of a resistant glacial till remnant that rises about 60 m above the pro-delta slope.

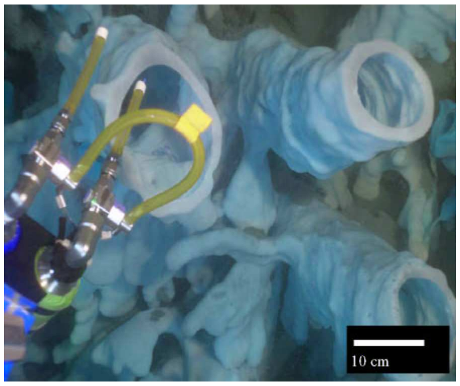


Figure 1: *In situ sampling of the water inhaled and exhaled by a glass sponge (A.vastus) using ROPOS robotic manipulator and the SIP water sampler.*

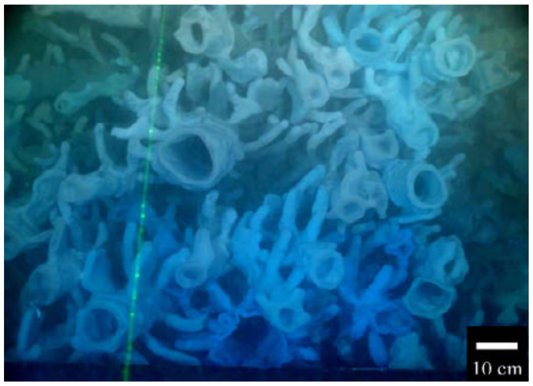


Figure 2: *The Fraser Ridge glass sponge reef, ROPOS dive November 2004. The reef is composed mainly of two sponge species, Aphrocallistes vastus and Heterochone calyx that form densely packed stands.*

Glass sponges are unique animals. Most of their tissue is made of giant, multinucleated cells called a syncytium. This body organization allows the sponge to transmit electrical signals over its nerveless body perhaps in the style of its metazoan ancestors. The sponge soft tissue forms a thin veneer over a rigid skeleton made of glass spicules that accounts for ~85% of the sponge mass. Interestingly, these "biological glass fibers" are similar to commercial optical fibers. The reef forming glass sponges are large animals reaching over 1.5 m high and a few metres wide. Their glass spicules are fused together so that when the sponge dies, its skeleton remains to form the

> *foundation for the settlement of new sponges. The growth of new generations over the skeletons of their progenitors is the process that creates the glass sponge reef in a similar fashion to the formation of a coral reef.*
>
> *Like other sponges, glass sponges are probably suspension feeders feeding by pumping large quantities of seawater via a specialized filtration system. Study of the diet composition, pumping activity and metabolism of glass sponge has so far been hampered by their deep, remote and inaccessible habitat. A group of scientists from PGC and the Universities of Alberta, Victoria and Washington, led by Dr. Sally Leys (U. Alberta), are gearing up to conduct the first in situ biological study of glass sponge reefs.*
>
> …
>
> *In the Jurassic period, glass sponge reefs covered substantial parts of the ancient Tethys Sea seafloor forming a 7000 long km reef system. This reef system is the largest known biological structures ever built on earth. Thought to be extinct with the dinosaurs, the discovery of extant glass sponge reefs in BC waters has opened up an exciting opportunity to look into the functioning of an ancient benthic community and its interaction with the surrounding ecosystem. The Strait of Georgia reefs include the high turbidity site at Fraser Ridge and the McCall Bank setting which resembles the northern reefs complex. These reefs are located in close vicinity to many of BC's major research facilities. VENUS instruments may be able to reach this site, depending on the final routing of the Strait of Georgia crossing.*

1.2. VENUS

The Victoria Environmental Network Under the Sea (VENUS) went operational on 8 February 2006. Some VENUS access points (nodes) were located in close proximity to the area where the glass sponge reefs were found so the plan was to deploy a sensor net on the reef and to connect to it via a VENUS node making it part of the VENUS system. The following extract from the VENUS project web site [27] provides a good overview of the VENUS project:

> *VENUS will be a network of instruments in the ocean to observe the marine waters around southern Vancouver Island in both Saanich Inlet and the Strait of Georgia. Measurements, images, and sound will be delivered to scientists, managers, the public, and a data archive via fibre-optic cables laid from two landfall sites. The cables will also deliver power for instruments, lights, and remotely operated vehicles. For the first time, we will not have to wait for data from periodically recovered instruments - it will be delivered immediately to researchers as events in the ocean unfold. The innovation of this array lies in three areas:*
>
> - *scientists will now be able to interact with the ocean through their instruments in real-time*

- *instruments will no longer be limited by power limitations, thus new designs can be supported*
- *a centralized data management and archive facility will interact with the researcher and the public, allowing data mining of long-term observations.*

The Straits of Georgia and ajoining waters around southern Vancouver Island are among the most heavily used bodies of water in Canada. Our network of sensors will return information on the behaviour of the marine environment and the seafloor. The 4 km array into Saanich Inlet from the Institute of Ocean Sciences in Pat Bay will focus on inlet renewal, ocean chemistry, biological interactions among species, and system design development of seafloor observatory components. A major line will cross the Strait of Georgia from near the Iona Causeway and will support instruments across the Strait to measure water properties, river dynamics, whale acoustics, fish migrations, plankton distributions, among other studies. A spur line along the Fraser Delta front will support a major project to monitor the stability of the Delta sediments, which are prone to collapse events.

The VENUS project will have a series of nodes located on the ocean floor. Individual instruments will either connect directly to a node or to an intermediate "Scientific Instrument Interface Module" (SIIM).

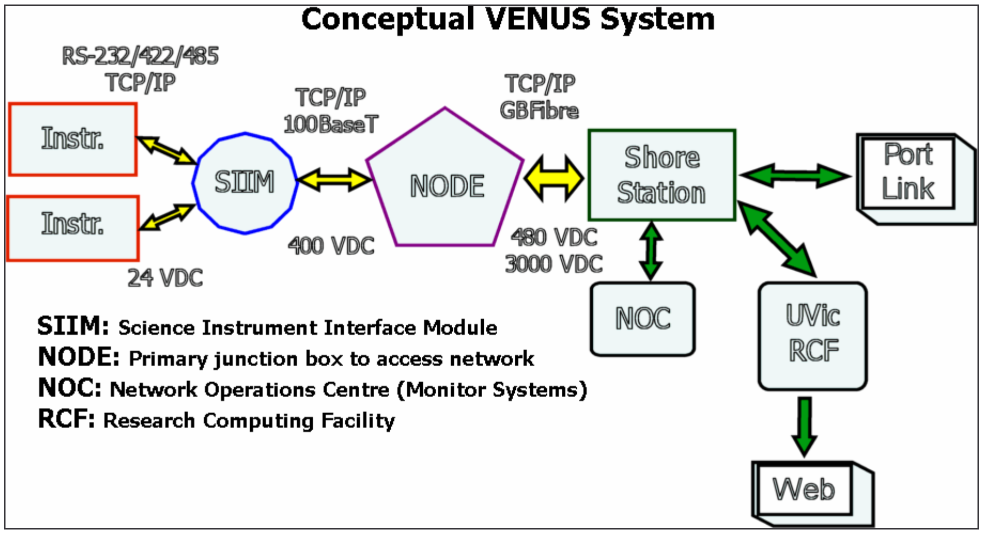

Conceptual VENUS System
RS-232/422/485
TCP/IP
Instr.
Instr.
24 VDC
SIIM
TCP/IP
100BaseT
400 VDC
NODE
TCP/IP
GBFibre
480 VDC
3000 VDC
Shore Station
Port Link
NOC
UVic RCF
Web
SIIM: Science Instrument Interface Module
NODE: Primary junction box to access network
NOC: Network Operations Centre (Monitor Systems)
RCF: Research Computing Facility

Figure 3: VENUS Architecture [26]

The node connection provides power and communication via a wet-mateable connector. The communication is 10/100 BaseT Ethernet and the power is 400VDC. Both power and communication is provided by a single connector. The connectors are wet-mateable so they can be connected to the node by an ROV. All connections on the SIIM and/or individual instruments will be dry-mateable. An instrument package will be completely assembled on the surface, deployed as a system and connected to the node by a single connector.

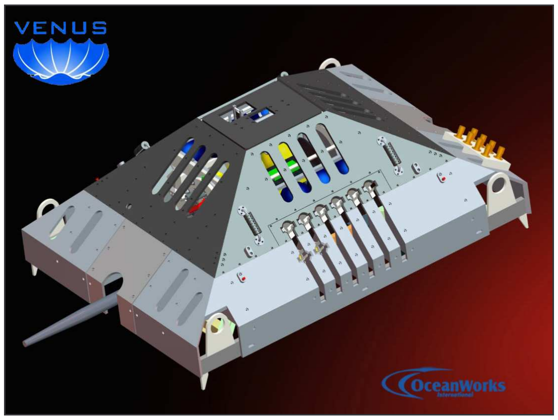


Figure 4: VENUS Node, Concept [26]

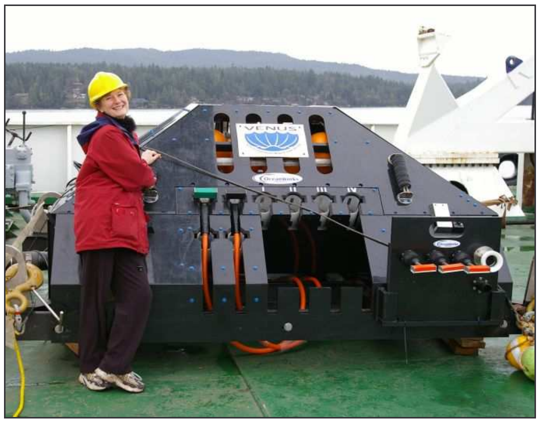


Figure 5: VENUS Node, Actual Device [26]

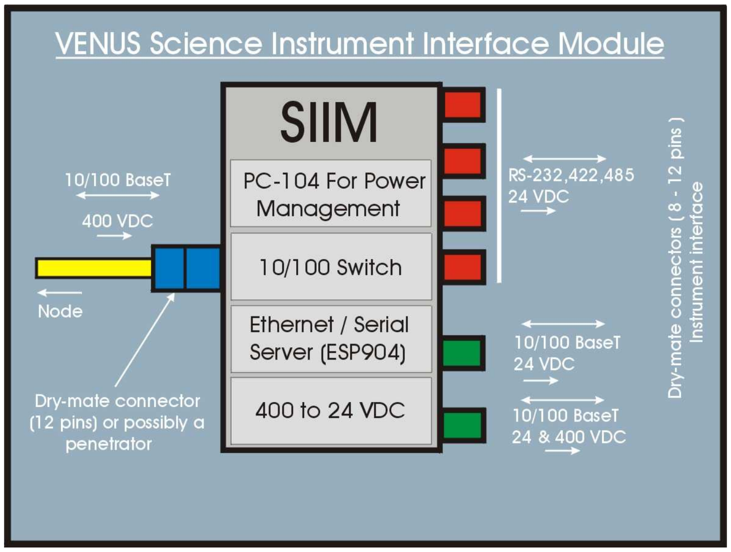

VENUS Science Instrument Interface Module
SIIM
PC-104 For Power Management
10/100 Switch
Ethernet / Serial Server (ESP904)
400 to 24 VDC
10/100 BaseT
400 VDC
Node
Dry-mate connector (12 pins) or possibly a penetrator
RS-232,422,485
24 VDC
10/100 BaseT
24 VDC
10/100 BaseT
24 & 400 VDC
Dry-mate connectors (8 - 12 pins)
Instrument interface

Figure 6: VENUS SIIM [26]

1.3. Embedded Computing and Sensor Nets

The ability to easily embed significant computing power is changing the way designers are approaching many problems. Tasks once considered too costly or simply not possible due to size, cost, or power restraints are now achievable and tremendously practicable

An embedded system is a special-purpose system in which the computer is completely encapsulated by the device it controls. As opposed to a general-purpose computer, such as a personal computer, an embedded system performs pre-defined tasks, usually with very specific requirements. In an application where the system is dedicated to a specific task, design engineers are able to optimize the system extensively, thus considerably reducing the size and cost of the end product. [31]

Embedded computing is a broad domain that encompasses both large and small systems. There are large general-purpose embedded systems with the processing power of personal computers and small application-specific embedded systems with comparatively little processing power.

Sensor nets are at the very small end of the embedded computing range. Sensor nets are comprised of small independent battery-operated devices that contain all the functional components needed to gather data about their local environment and to process, store, and form a network with other devices to communicate that data to some central location.

Several names are being used to refer to the individual sensor net devices and there is no universally accepted term. The term “node” is a very general term for any device on a network and applies to sensor nets as well. The term “mote” is often used to refer to sensor net nodes but it is used exclusively to refer to sensor net nodes that use radio-frequency (RF) to communicate. The devices used to monitor glass sponge reefs must function underwater and RF will not work underwater. Mote is therefore an inappropriate term and the more generic term of node will be used in this paper.

A Sensor net node is a single board computer. “Single board computers (SBCs) are complete computers built on a single circuit board.” [31] The term SBC can mean large boards, such as those used in typical desktop PCs, but large boards are more often called “mother boards” and the term SBC is usually reserved for smaller boards intended for embedded applications. Large SBC’s generally use separate chips for the central processing unit (CPU), memory and input/output (I/O), but the small SBCs used for sensor net nodes are powered by microcontrollers (uC’s).

A uC is a single chip that contains a CPU, memory for both program and data storage, and I/O. The processing speeds, memory sizes, and I/O capacity of uC’s vary tremendously. Twenty-three different manufacturers [31] are producing microcontrollers that continue to get smaller, cheaper and include more features. The Freescale Semiconductor, ColdFire 5270 microcontroller, is an example of the rapidly increasing level of integration. Along with all the other features found on most other microcontrollers the 5270 also includes a 10/100 Ethernet controller. The range of products currently available is both a blessing and a curse. It gives the design engineer tremendous freedom to select the device that best suits the application, but it also requires knowledge of a wide range of products with very different features that, as will be shown, makes comparison a complex task.

The glass sponge reef sensor net must operate underwater. Two possible underwater communication mediums are sound and light; acoustical and optical. Because of concerns that sound may adversely affect local marine life optical communications was chosen. The optical communications system was in development by others so no details were available.

2. Related Research

Several underwater sensor network projects had been completed or were ongoing but (like VENUS) they all used very large nodes that were connected to, and often powered by, cable tethers to surface buoys. [1][2][20][27].

The wireless RF land-based sensor networks were much closer to the type of platform needed. The RUNES paper on “Existing Architectures and Components” [7] identified that the smallest platform was probably “Smart Dust” (shown below), and the most popular platforms were the “mote” series developed at UC Berkeley.

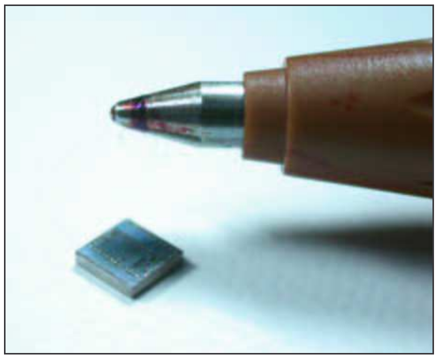
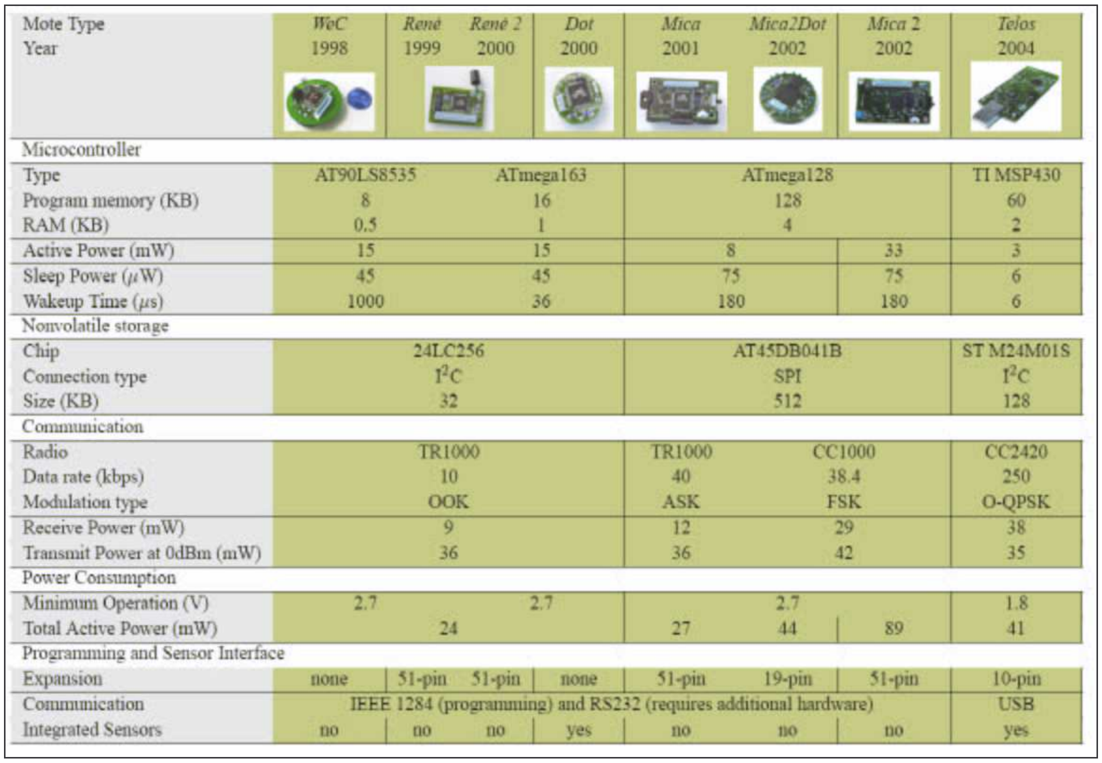


Figure 7: Smart Dust [7]

> *It should be mentioned that the reason for singling out the MOTE platform in particular lies mainly in its huge popularity. It is by far the most commonly used sensor network building block, and thus the baseline other platforms are compared to, and which should be used for early prototyping to make the results comparable with other research initiatives all around the world.* [7]

They also did a comparative summary of the key features of each of the Berkeley mote versions as shown below.



Mote Type	*WeC*	*René*	*René 2*	*Dot*	*Mica*	*Mica2Dot*	*Mica 2*	*Telos*
Year	1998	1999	2000	2000	2001	2002	2002	2004
Microcontroller								
Type	AT90LS8535		ATmega163		ATmega128			TI MSP430
Program memory (KB)	8		16		128			60
RAM (KB)	0.5		1		4			2
Active Power (mW)	15		15		8		33	3
Sleep Power (μW)	45		45		75		75	6
Wakeup Time (μs)	1000		36		180		180	6
Nonvolatile storage								
Chip	24LC256				AT45DB041B			ST M24M01S
Connection type	I^2C				SPI			I^2C
Size (KB)	32				512			128
Communication								
Radio	TR1000				TR1000	CC1000		CC2420
Data rate (kbps)	10				40	38.4		250
Modulation type	OOK				ASK	FSK		O-QPSK
Receive Power (mW)	9				12	29		38
Transmit Power at 0dBm (mW)	36				36	42		35
Power Consumption								
Minimum Operation (V)	2.7		2.7		2.7			1.8
Total Active Power (mW)	24				27	44	89	41
Programming and Sensor Interface								
Expansion	none	51-pin	51-pin	none	51-pin	19-pin	51-pin	10-pin
Communication	IEEE 1284 (programming) and RS232 (requires additional hardware)							USB
Integrated Sensors	no	no	no	yes	no	no	no	yes

Figure 8: The Berkeley Motes [7]

The most recent mote, the Telos, is an open design and it is available (in slightly different flavours) from two different manufacturers:

- From Crossbow as the "TelosB"
- From Moteiv as the "Tmote Sky"

Both of these products were surveyed to evaluate their suitability and were rejected for two main reasons:

- They were intended for land-based applications using RF communication and the RF circuits were a major part of the platform.
- They did not have any Ethernet capability.[*]

Without the RF circuits the Telos mote was basically the MSP430 uC and memory. As will be seen in the next section, the MSP430 was available on more suitable platforms.

The investigation of the Telos motes was very helpful in establishing a reference for how small the uC and memory could be. If the Telos was a fully functioning node for a RF-based sensor net, then all that was needed, in principle, for an optical sensor net was to replace the RF circuits with optical circuits.

3. SBC for a Sensor Net Node

The required SBC features were identified then the market was surveyed for suitable candidates. The specific features of ten of the most suitable SBC's were compared and the most suitable SBC was selected for detailed evaluation.

3.1. SBC Features

The investigation of the Telos motes established that a microcontroller with the processing and memory capacity of the MSP430 would work but the specific features needed for an SBC that would be used to create an underwater sensor net node were not well defined.

> *We begin by discussing various criteria, based on which EmNet solutions can be compared and evaluated. Naturally, very little absolutes exist in terms of these criteria. A solution completely inappropriate for one application can be very practicable for another.* [7]

Two must-have features and several should-have features were identified.

The candidate SBC's had to have the following features:

- Ethernet Interface
- HTTP, TCP/IP support

The most suitable SBC had to have the best combination of:

- Smallest: Power, Size, and Cost
- Biggest: Memory, Analog and Digital I/O, Serial Interfaces

[*] This was considered an essential feature as explained in the next section.

- Expansion Capability
- Open Design and Real-Time Clock
- OS, C Compiler and Integrated Development Environment

3.1.1. VENUS Compatibility

The sensor net had to be able to become part of the VENUS system. A key aspect of the VENUS concept was that all instruments would be manageable, and the data they collected would be available, via a web interface. The VENUS system was also Ethernet based, and since some sensor net nodes had to connect to the VENUS system, either a special node with Ethernet capability was required or all nodes needed to have it. One node type was the simpler approach and because Ethernet capability was relatively common and inexpensive a single node type with an Ethernet interface was not only preferable but quite practicable. The selected SBC therefore had to have an Ethernet interface and support for TCP/IP and HTTP.

The VENUS system used PC104 products so they were investigated to see if any existed that might suit this application. These boards had a great deal of capacity but they were also larger, more power hungry, and more expensive than any of the competing SBC's. The Technologic Systems, TS-7260 was one of the very few exceptions as the comparison data showed. Most PC104 boards were more like the Micro/Sys SBC1491 (announced on 19 April 2006, [15]) which had a power consumption of 12,000mW; several orders of magnitude greater than the other devices considered. These products were generally not suitable for embedded very low-power battery-operated applications but they were included in the survey and two were included in the comparison for reference.

3.1.2. Power, Size and Cost (Smaller is Better)

Power consumption was a critical selection factor. The sensor net nodes would be battery powered and deployed on the ocean floor. Retrieval to change batteries would be a difficult and costly operation. The entire node needed to be small and low cost so large battery packs were not an option. SBC's with lower power consumption were not only preferable but a priority.

There were numerous factors that affected power consumption: most SBC's had LED status indicators that could be disconnected or disabled; many uC's and Ethernet controllers had low power modes; most uC's could operate at different clock speeds. Because SBC power consumption included many things that could be disabled SBC power by itself was not a sufficient indicator of suitability. A key factor in SBC power consumption was the uC that it used. To get a better sense of which SBC had better power consumption ratings two power consumption values were identified:

- The power consumed by just the microcontroller running at the speed specified for the SBC.

- The power consumption for the entire SBC*.

Not all of the surveyed SBC's ran at 5V and the on-board voltage regulation allowed a range of supply voltages. To provide meaningful comparisons all SBC power ratings were calculated using the supply current and the system voltage not the supply voltage. If, for example, an SBC actually ran at 5V but was supplied by a 10V source and the current rating was 100mA the power would appear to be 1000mW when it was really 500mW. Similarly all uC power ratings were calculated using the typical current of the uC at 5V. To further simplify comparison the power values were listed in mW not as separate voltages and currents.

Physical size and cost were issues and SBC's that were smaller and less expensive were preferable. These two factors tended to change in proportion with power so, in most cases, the lower power device was also the smaller and less expensive.

Because most products had lower costs for higher volumes only the cost of purchasing one unit was used.

3.1.3. Memory, I/O, and Expansion (Bigger is Better)

The SBC needed enough memory available to support the essential functions and leave space for future development. The Telos mote provided a reference to how small memory could be but bigger was better. Several products had both Flash and EEPROM and each was used for different purposes so they were listed separately. Some systems had the ability to add memory so, for comparison, only the base memory capacity was used.

There were several unknown interfacing requirements. The node needed to interface with environmental sensor(s) and the (yet to be defined) optical communication system. Sensor interfaces were typically analog or digital and the optical communications interface would likely be digital (parallel) or serial.

In addition the SBC needed to be expandable so that connecting to different peripheral devices was not just possible but relatively simple. Some SBC's had connectors that made connection to expansion devices easy, others did not.

In summary SBC's with more memory, analog and digital I/O, serial interfaces and expansion capability were preferable.

3.1.4. Open Design and RTC (Should Have)

Two criteria were that were considered nice to have, or should have features were an open design and a real-time clock.

Some SBC's were proprietary designs and some were open designs. An open design meant that development could be done with the SBC and a custom board could also be

* If available, as some power consumption values were not available.

created, if desired, that used some or all of the same circuits and software. The open design was clearly preferable.

The requirement to operate independently and gather data even when not connected implied that each node should have a real time clock (RTC). This function could be added to any of the SBC's but if already present would have made the product a better choice.

3.1.5. Operating Systems

An OS was not mandatory. Many small low-power embedded devices run simple cooperative multitasking such as the TCP/IP stacks from Texas Instruments and Microchip.

Embedded computing in general and sensor net nodes in particular require a different approach than general-purpose computing.

> *The principal role of embedded software is interaction with the physical world. Consequently, the designer of that software should be the person who best understands that physical world. The challenge to computer scientists, should they choose to accept it, is to invent better abstractions for that domain expert to do her job.*
>
> *…The fact is that the best-of-class methods offered by computer scientists today are, for the most part, a poor match to the requirements of embedded systems.* [13]

A recent survey of operating system use in embedded systems [29], identified that about 36% either use no OS or use an internally developed one.

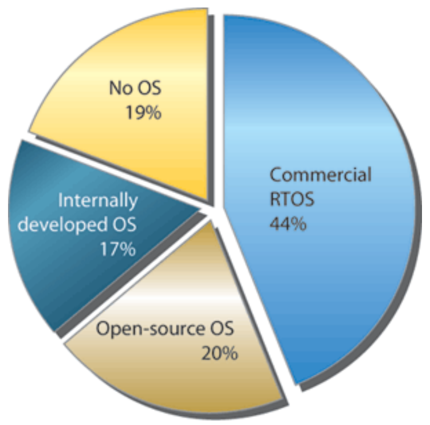


Figure 9: OS Use in Embedded Systems [29]

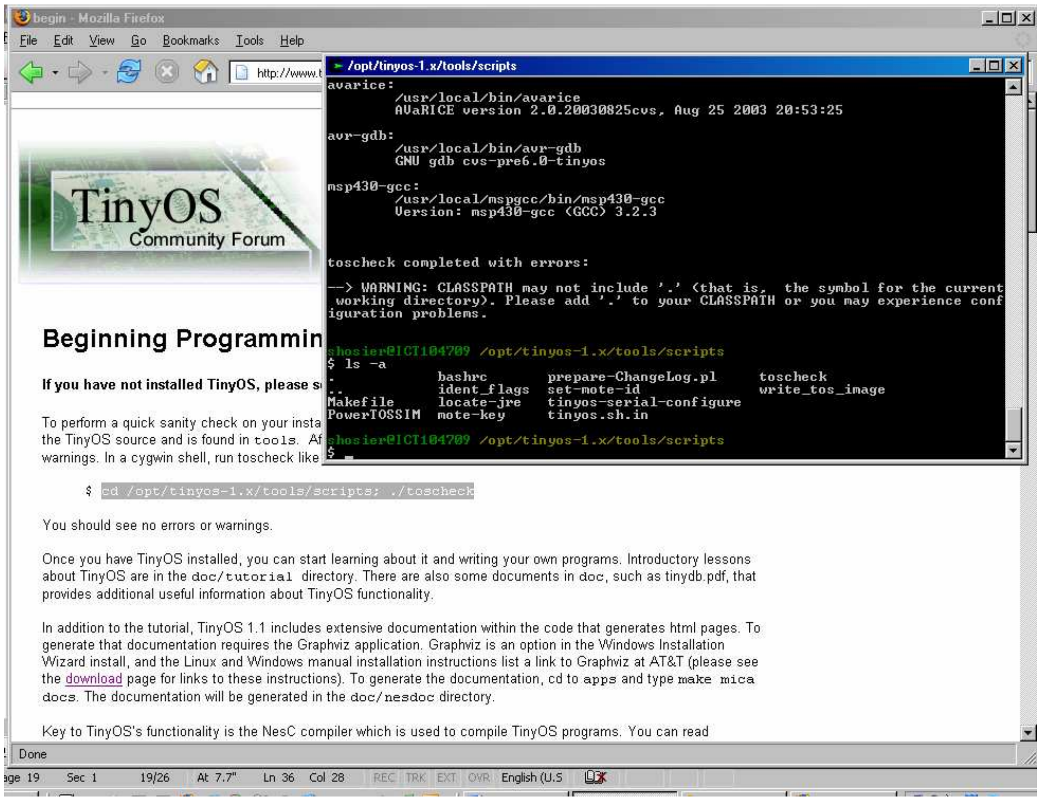
Low-power embedded systems have very specific tasks to perform. An OS is often not used because the convenience offered by an operating system is not needed and because very tight resource restrictions, such as memory and power, usually preclude it.

However an OS may be desired and development may be easier with an OS, so the availability of suitable OS's was investigated. Many OS's were available but most were not suitable for very low-power applications. Three that were specifically developed for low-power embedded applications were TinyOS, FreeRTOS, and SALVO. The first two were open, the third was proprietary.

A group at UC Berkeley, in collaboration with Crossbow Technology Inc., developed TinyOS.

> *TinyOS is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. TinyOS's component library includes network protocols, distributed services, sensor drivers, and data acquisition tools – all of which can be used as-is or be further refined for a custom application. TinyOS's event-driven execution model enables fine-grained power management yet allows the scheduling flexibility made necessary by the unpredictable nature of wireless communication and physical world interfaces.* [28]

TinyOS was downloaded and installed in a Windows XP environment. The overall initial impression was not very positive. There does not seem to be much current development. There were numerous bugs. The installation was complex with several warnings about required versions of TinyOS and Cygwin, and required the use of an installation manual to ensure correct options were chosen at each step. The download and installation took several hours; partly because of difficulty resolving what needed to be installed and where to get it; and partly because of poorly written documentation. The specified post-installation check indicated potential errors as shown in the screen capture below.


begin - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.t
TinyOS
Community Forum
/opt/tinyos-1.x/tools/scripts

```
avarice:
        /usr/local/bin/avarice
        AVaRICE version 2.0.20030825cvs, Aug 25 2003 20:53:25

avr-gdb:
        /usr/local/bin/avr-gdb
        GNU gdb cvs-pre6.0-tinyos

msp430-gcc:
        /usr/local/mspgcc/bin/msp430-gcc
        Version: msp430-gcc (GCC) 3.2.3



toscheck completed with errors:

--> WARNING: CLASSPATH may not include '.' (that is,  the symbol for the current
 working directory). Please add '.' to your CLASSPATH or you may experience conf
iguration problems.


shosier@ICT104709 /opt/tinyos-1.x/tools/scripts
$ ls -a
.           bashrc       prepare-ChangeLog.pl     toscheck
..          ident_flags  set-mote-id              write_tos_image
Makefile    locate-jre   tinyos-serial-configure
PowerTOSSIM mote-key     tinyos.sh.in

shosier@ICT104709 /opt/tinyos-1.x/tools/scripts
$ _
```

Beginning Programmin

If you have not installed TinyOS, please s

To perform a quick sanity check on your insta
the TinyOS source and is found in `tools`. Af
warnings. In a cygwin shell, run toscheck like

```
$ cd /opt/tinyos-1.x/tools/scripts; ./toscheck
```

You should see no errors or warnings.

Once you have TinyOS installed, you can start learning about it and writing your own programs. Introductory lessons about TinyOS are in the `doc/tutorial` directory. There are also some documents in `doc`, such as tinydb.pdf, that provides additional useful information about TinyOS functionality.

In addition to the tutorial, TinyOS 1.1 includes extensive documentation within the code that generates html pages. To generate that documentation requires the Graphviz application. Graphviz is an option in the Windows Installation Wizard install, and the Linux and Windows manual installation instructions list a link to Graphviz at AT&T (please see the download page for links to these instructions). To generate the documentation, cd to `apps` and type `make mica docs`. The documentation will be generated in the `doc/nesdoc` directory.

Key to TinyOS's functionality is the NesC compiler which is used to compile TinyOS programs. You can read

Done

age 19 Sec 1 19/26 At 7.7" Ln 36 Col 28 REC TRK EXT OVR English (U.S

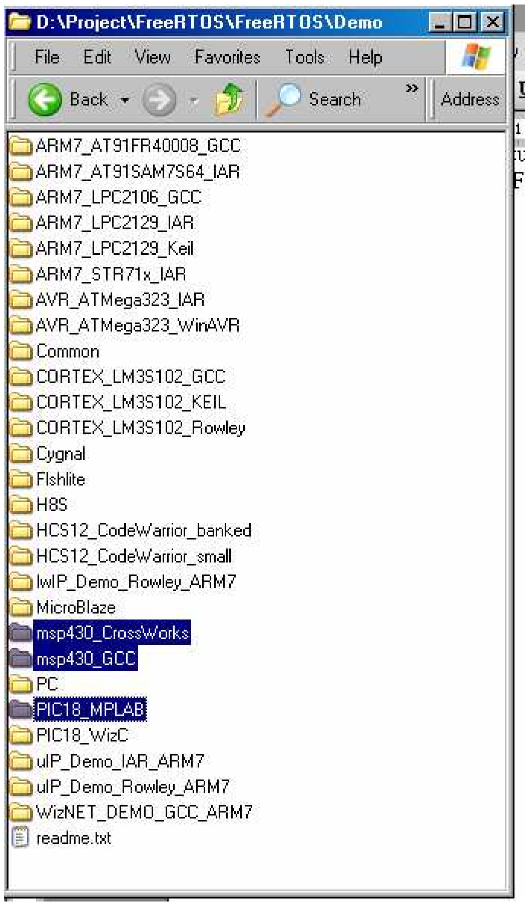
Figure 10: TinyOS Installation

TinyOS did seem to have numerous research projects ongoing and the negative impression created by difficulty downloading and installing TinyOS may not be reflective of the product in general. However two additional factors did strongly indicate that it was not a good solution:

- The programming language required by TinyOS was a non-standard variant of C called Nes-C (Nested C) [10][14] and it only worked with TinyOS.
- The only hardware platforms listed were those from Crossbow that use the MSP430 uC. The general impression was that this product was more of a proprietary product created for Crossbow.

The second OS specifically developed for low-power embedded use was FreeRTOS. This was an open source, pre-emptive and cooperative, real-time[*] operating system. This was a more active product. The platform-independent source code with demos was a folder that downloaded and installed quite easily in a few minutes. It had been ported to many systems and demos were available for all of them as shown in the screen capture below. The top two uC's identified by the survey, the MSP430 and the PIC18, were both supported by this OS.

[*] Real-time is actually a very bad term. Analog devices are real-time, uC's are not. The term refers more to the ability to respond in a predictable and acceptable time frame.


D:\Project\FreeRTOS\FreeRTOS\Demo
File Edit View Favorites Tools Help
Back
Search
Address

ARM7_AT91FR40008_GCC
ARM7_AT91SAM7S64_IAR
ARM7_LPC2106_GCC
ARM7_LPC2129_IAR
ARM7_LPC2129_Keil
ARM7_STR71x_IAR
AVR_ATMega323_IAR
AVR_ATMega323_WinAVR
Common
CORTEX_LM3S102_GCC
CORTEX_LM3S102_KEIL
CORTEX_LM3S102_Rowley
Cygnal
Flshlite
H8S
HCS12_CodeWarrior_banked
HCS12_CodeWarrior_small
lwIP_Demo_Rowley_ARM7
MicroBlaze
msp430_CrossWorks
msp430_GCC
PC
PIC18_MPLAB
PIC18_WizC
uIP_Demo_IAR_ARM7
uIP_Demo_Rowley_ARM7
WizNET_DEMO_GCC_ARM7
readme.txt

Figure 11: FreeRTOS Demo Folder

Salvo was the third low-power embedded OS. A commercial offering that retailed for $1250US[*].

> *Salvo™ is the first Real-Time Operating System (RTOS) designed expressly for very-low-cost embedded systems with severely limited ROM and RAM. Typical applications use 1-2K ROM and 50-100 bytes of RAM. By bringing real-time multitasking to the highest-volume end of the embedded processor market, Salvo gives you the power to quickly create low-cost, smart and sophisticated products for the Internet-enabled post-PC age. Pumpkin™ has currently certified Salvo for use with:*
>
> - *8051 family and its derivatives*
> - *ARClite microRISC synthesizeable 8-bit core*
> - *ARM® ARM7TDMI®*
> - *Atmel® AVR® and MegaAVR™*
> - *Motorola M68HC11*
> - *TI's MSP430 Ultra-Low Power Microcontroller*
> - *Microchip PIC12\14000\16\17\18 PICmicro® MCUs*
> - *Microchip PIC24 MCUs and dsPIC® DSCs*

[*] Educational pricing was only available on request.

- *TI's TMS320C2000 DSPs*

A free demo version was available but there were separate demo downloads for each target platform. The demo for the PIC18 was downloaded and installed easily in a few minutes. Overall impression was very positive. If the educational pricing is reasonable this may be a good choice.

As stated previously an OS was not mandatory, but it may be desirable and may facilitate development, so SBC's that had OS's available were considered preferable and more so if both proprietary and open OS's were available.

3.1.6. C Compiler and IDE

While there were many possible languages the two main ones were Java and C. There were many varieties of Java for small real-time embedded systems including JITS, JINI, TINY, and J2ME. Java had many aspects that could make it a much nicer programming language than C but Java was not the best choice for real-time applications and definitely not for very low-power real-time embedded systems. The severely limited resources demanded by the low power requirement made the use of C preferable over Java.

> *…their skepticism is well warranted. They see Java programs stalling for one third of a second to perform garbage collection and update the user interface, and they envision airplanes falling out of the sky. The fact is that the best-of-class methods offered by computer scientists today are, for the most part, a poor match to the requirements of embedded systems.* [13]

C was the most appropriate language; TinyOS, FreeRTOS, Salvo and the TI and Microchip TCP/IP stacks were all written in C. SBC's that had both free and commercial versions of standard C compilers available were more preferable.

The development environment was an important factor, but it was not possible to quantify it so the criterion was that there should have been an integrated development environment (IDE) available and preferably it should have been free.

3.2. SBC Comparison

Many different products were surveyed. Ten SBC's were chosen that were representative of the range of available products. All of the chosen products had all of the required features and a variety of the optional features. All of the manufacturers represented in the comparison produce many other similar products. Only one product was selected per manufacturer with the exception of Olimex which had two products in the comparison because they were from different uC families.

Appendix A contains the complete list of products surveyed.

The ten SBC's selected for comparison, in alphabetical order, were:
- Digi, DigiConnect EM
- Ethernut, 2.1

- Micro/Sys, SBC1491
- Modtronix, SBC65EC
- Netburner, Mod5270
- Olimex, LPC-E2124
- Olimex, MSP430-easyWeb3
- Rabbit Semiconductor, RCM3750 Rabbit Core
- SoftBaugh, TCP430
- Technologic Systems, TS-7260

3.2.1. Digi, DigiConnect EM [4]

uC: Net Silicon NS7520 **System Power:** 1,350mW **uC Power:** 508mW **RAM:** 8,000KB **Flash:** 4,000KB **EEPROM:** none **Serial:** RS232, SPI **RTC:** No **Analog I/O:** 8 **Digital I/O:** same 8 (lines can be either) **Expansion:** No **Size:** 49 x 40mm, 1960mm^2 **Open design:** No **OS's:** Proprietary (ThreadX) **C Compilers:** Proprietary **IDE's:** Proprietary (Microcross) **Cost:** $100 US	**Figure 12: DigiConnect EM**

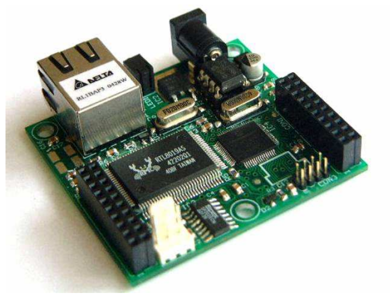
3.2.2. **Ethernut 2.1** [8]

uC: Atmel ATmega128 **System Power:** 1,500mW **uC Power:** 250mW **RAM:** 512KB **Flash:** 640KB **EEPROM:** 4KB **Serial:** RS232, RS485 **RTC:** No **Analog I/O:** 8 **Digital I/O:** 22 **Expansion:** Yes **Size:** 98 x 78mm, 7644mm^2 **Open Design:** No **OS's:** Proprietary (NutOS) **C Compiler's:** Proprietary and Open **IDE's:** Proprietary (ImageCraft) **Cost:** $200 US	Figure 13: Ethernut 2.1

3.2.3. **Micro/Sys, SBC1491** [16]

uC: ST Microelectronics, Atlas **System Power:** 9,500mW **uC Power:** 2,000mW **RAM:** 64,000KB **Flash:** 1,000KB **EEPROM:** none **Serial:** RS232 (x2) **RTC:** Optional **Analog I/O:** none (PC104 expansion module) **Digital I/O:** none (PC104 expansion module) **Expansion:** Yes (PC104 standard) **Size:** 90 x 96mm, 8640mm^2 **Open design:** No **OS's:** DOS, Linux, CE, NT, RTOS **C Compilers:** Proprietary and Open **IDE's:** Numerous for PC104 **Cost:** $455 US [15]	Figure 14: SBC1491

3.2.4. Modtronix SBC65EC [18]

uC: PIC18F6621 **System Power:** 290mW **uC Power:** 20mW **RAM:** 4KB **Flash:** 98KB **EEPROM:** 64KB **Serial:** RS323, SPI, I2C **RTC:** No **Analog I/O:** 12 **Digital I/O:** 32 **Expansion:** Yes **Size:** 59 x 54mm, 3186mm^2 **Open design:** Yes **OS's:** Proprietary and Open **C Compilers:** Proprietary and Open **IDE's:** Proprietary and Open **Cost:** $70 US	 Figure 15: SBC65EC

3.2.5. Netburner Mod5270 [19]

uC: ColdFire 5270 **System Power:** 2,255mW **uC Power:** Not specified **RAM:** 2000KB **Flash:** 512KB **EEPROM:** none **Serial:** RS323 (x3), I2C **RTC:** No **Analog I/O:** None **Digital I/O:** 47 **Expansion:** Yes **Size:** 51 x 66mm, 3366mm^2 **Open design:** No **OS's:** Proprietary (Netburner RTOS) **C Compilers:** Proprietary (Netburner) **IDE's:** Proprietary (Netburner) **Cost:** $150 US	Figure 16: Mod5270

3.2.6. Olimex, LPC-E2124 [21]

uC: ARM7TDMI-S **System Power:** Not Specified **uC Power:** 280mW **RAM:** 16KB **Flash:** 320KB **EEPROM:** none **Serial:** RS232 (x2), SPI, I2C **RTC:** Yes **Analog I/O:** 4 **Digital I/O:** 10 **Expansion:** Yes **Size:** 80 x 90mm, 7200mm^2 **Open design:** Yes **OS's:** Proprietary and Open **C Compilers:** Proprietary and Open **IDE's:** Proprietary and Open **Cost:** $100 US	**Figure 17: LPC-E2124**

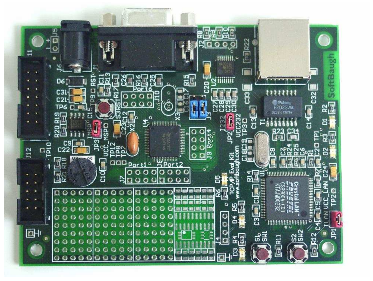
3.2.7. Olimex, MSP430-easyWeb3 [21]

uC: MSP430F149 **System Power:** Not Specified **uC Power:** 30mW **RAM:** 2KB **Flash:** 60KB **EEPROM:** none **Serial:** RS232 **RTC:** No **Analog I/O:** 10 **Digital I/O:** 48 **Expansion:** Yes **Size:** 76 x 60mm, 4560mm^2 **Open design:** Yes **OS's:** Proprietary and Open **C Compilers:** Proprietary and Open **IDE's:** Proprietary and Open **Cost:** $70 US	**Figure 18: MSP430-easyWeb3**

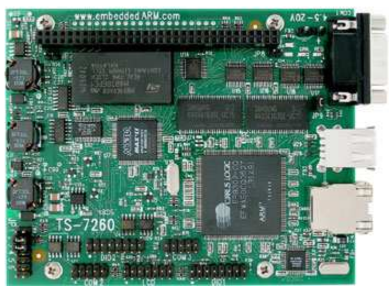
3.2.8. Rabbit Semiconductor, RCM3750 Rabbit Core [23]

uC: Rabbit 3000 **System Power:** 875mW **uC Power:** Not Specified **RAM:** 512KB **Flash:** 1,512KB **EEPROM:** none **Serial:** RS232, IrDA, SPI **RTC:** Yes **Analog I/O:** None **Digital I/O:** 33 **Expansion:** Yes (connector on bottom of board) **Size:** 75 x 30mm, 2250mm^2 **Open design:** No **OS's:** Proprietary **C Compilers:** No **IDE's:** Proprietary **Cost:** $75	Figure 19: RCM3750

3.2.9. SoftBaugh, TCP430 [24]

uC: MSP430F1611 **System Power:** Not specified **uC Power:** 30mW **RAM:** 10KB **Flash:** 48KB **EEPROM:** none **Serial:** RS232 **RTC:** No **Analog I/O:** 1 **Digital I/O:** 48 **Expansion:** Yes **Size:** Not Specified **Open design:** No **OS's:** Proprietary and Open **C Compilers:** Proprietary and Open **IDE's:** Proprietary and Open **Cost:** $250 US	 Figure 20: TCP430

3.2.10. Technologic Systems, TS-7260 [25]

uC: ARM920-200MHz **System Power:** 1000mW **uC Power:** 160mW **RAM:** 32000KB **Flash:** 32000KB **EEPROM:** none **Serial:** RS232 (x3), SPI, USB (x2) **RTC:** Optional **Analog I/O:** 2 **Digital I/O:** 30 **Expansion:** Yes **Size:** 97 x 115mm, 11155mm^2 **Open design:** No **OS's:** DOS, Linux **C Compilers:** Proprietary and Open **IDE's:** Proprietary and Open **Cost:** $179 US	 **Figure 21: TS-7260**

3.3. SBC Selection

Power consumption had been identified as a critical factor for this application. The table below shows the ten SBC's ranked in approximate order of lowest to highest power consumption. Power consumption tended to correlate with memory size with the notable exception of the TS-7260.

Table 1: Power Consumption Ranking

SBC	SBC Power (mW)	uC Power (mW)	RAM (KB)	Flash (KB)	EEPROM (KB)	Cost ($US)
SBC65EC	290	20	4	98	64	$70
TCP430	N.S.	30	10	48	None	$250
MSP430	N.S.	30	2	60	None	$70
RCM3750	875	N.S.	512	1,512	None	$75
TS-7260	1,000	160	32,000	32,000	None	$179
LPC-E2124	N.S.	280	16	320	None	$100
Ethernut 2.1	1,500	250	512	640	4	$200
Digi EM	1,350	508	8,000	4,000	None	$100
Mod5270	2,255	N.S.	2,000	512	None	$150
SBC1491	9,500	2,000	64,000	1,000	None	$455

If power consumption had not been a major factor the Technologic Systems, TS-7260 would have been a excellent choice. It had very low power consumption considering the tremendous amount of memory available. It could run Linux, it had excellent development support, an onboard RTC, PC104 compatibility and all of this at a very good price. This board was not chosen as the most suitable for this application but may be a good candidate for others.

As power consumption was a major factor the Modtronix SBC65EC, SoftBaugh TCP430, and Olimex MSP430-easyWeb3 were better choices than any of the others. All three had much lower power consumption but the price on the TCP430 was much too high so it was eliminated leaving only the SBC65EC and MSP430.

The top two SBC's were compared in terms of the other selection parameters in the table below. As shown, there were no noticeable differences regarding these factors; the two boards were essentially the same.

Table 2: SBC65EC and MSP430 Comparison

	SBC65EC	**MSP430**
Serial	RS232 (x2), SPI, I2C	RS232 (x2), SPI
RTC	No.	No
Analog I/O	12	13
Digital I/O	32	48
Expansion	Yes	Yes
Size	3,186mm^2	4,560mm^2
Open Design	Yes	Yes
OS's	Proprietary and Open	Proprietary and Open
C Compilers	Proprietary and Open	Proprietary and Open
IDE's	Proprietary and Open	Proprietary and Open

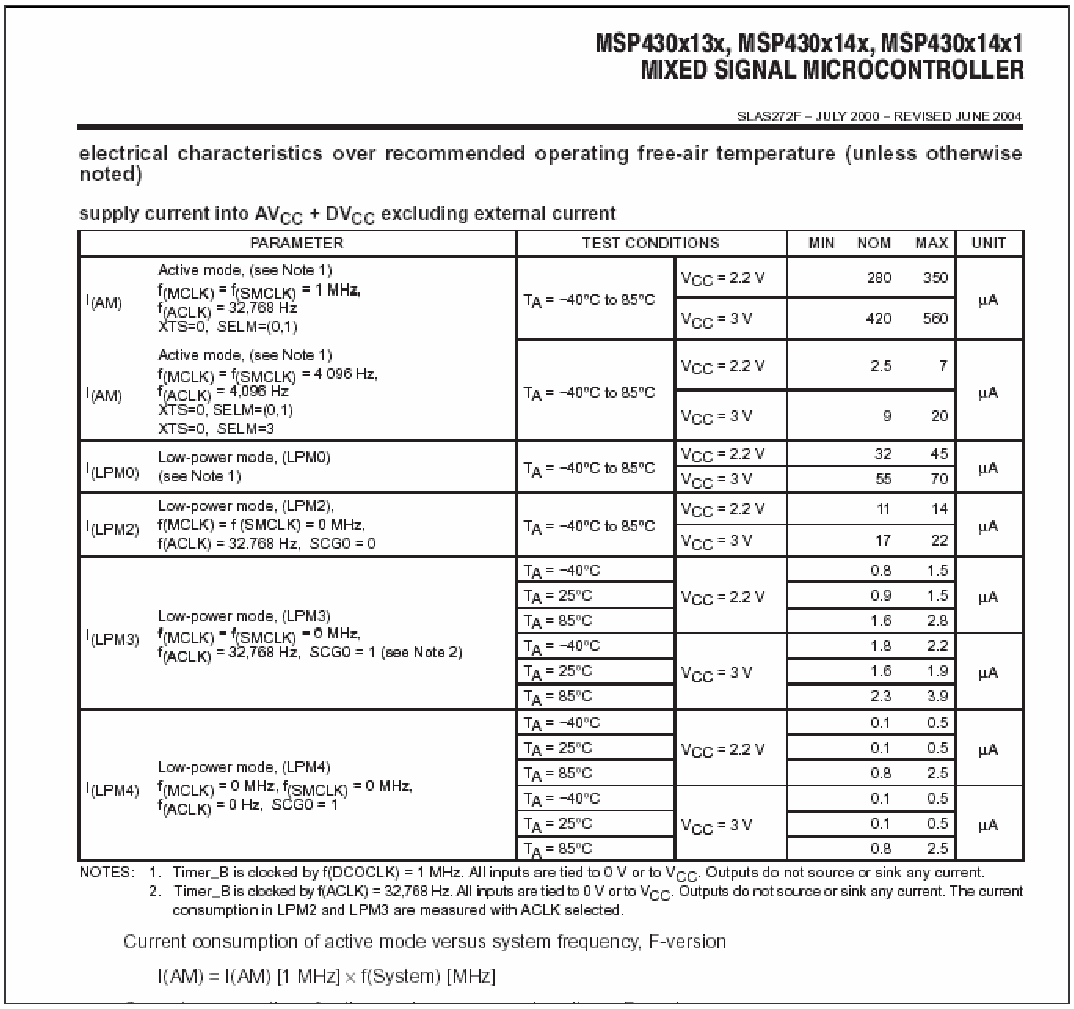
Since microcontroller performance was a key factor, the detailed ratings for the uC's used on these two boards were compared. The MSP430 board uses the Texas Instruments MSP430F149 and the Modtronix SBC65EC board uses the Microchip 18F6621. As shown in the table below, these two devices were amazingly close in all parameters.

Table 3: MSP430F149 and PIC18F6621 Comparison

		MSP430F149	**PIC18F6621**
Manufacturer		Texas Instruments	Microchip
Power [*]	I Running	4mA	4mA
	I Sleeping	1uA	1uA
	Modes	Yes	Yes
	Supply Voltage	2.7 to 3.6V	2 to 5.5V
Memory	Flash	60 KB	64KB
	RAM	2KB	4KB[†]
	EEPROM	None	1KB
I/O	A/D	8 x 12bit	13 x 10Bit
	Comparators	1	2
	Timers	2 x 16bit	3 x 16bit
	Digital I/O	48	32
	Serial	USART, SPI	USART, SPI, I2C
CLK	Range	DC to 8 MHz	DC to 40 MHz
	Used on SBC	8 MHz	20 MHz
ICSP[‡]		Yes	Yes
Cost[§]		$10.11US	$8.31US

[*] Power consumption varied greatly and depended on many factors. It will be analyzed more completely in the next section. The values listed are representative of typical ranges.
[†] The PIC18F actually has 3936 bytes of SRAM but for consistency it is rounded to 4KB
[‡] ICSP = In-Circuit Serial Programming
[§] See Appendix B for pricing references.

Most of the values used in the above table were extracted directly from the respective data sheets but supply current was more difficult. Supply current varies with many different parameters and two major ones were supply voltage and frequency. A problem in comparing the two devices was that they had different voltage and frequency ranges. The PIC18F6621 went from DC to 40MHz and ran at 40MHz on the SBC65EC board. The MSP430F149 went from DC to 8MHz and ran at 8MHz on the MSP430 board. There was no good way to decide what voltage or frequency to use. The MSP430 may have performed the same number of tasks per second at 3V and 8MHz as the PIC18F6621 did at 4V and 40MHz. For simplicity the comparison was done with both devices operating at the same voltage and clock speed. They both operated at 8MHz so that was used as the frequency parameter, and both devices operated at 3V so that was used for the voltage parameter.



MSP430x13x, MSP430x14x, MSP430x14x1
MIXED SIGNAL MICROCONTROLLER

SLAS272F – JULY 2000 – REVISED JUNE 2004

electrical characteristics over recommended operating free-air temperature (unless otherwise noted)

supply current into AV_{CC} + DV_{CC} excluding external current

PARAMETER		TEST CONDITIONS		MIN	NOM	MAX	UNIT
$I_{(AM)}$	Active mode, (see Note 1) $f_{(MCLK)} = f_{(SMCLK)}$ = 1 MHz, $f_{(ACLK)}$ = 32,768 Hz XTS=0, SELM=(0,1)	T_A = −40°C to 85°C	V_{CC} = 2.2 V		280	350	µA
			V_{CC} = 3 V		420	560	
$I_{(AM)}$	Active mode, (see Note 1) $f_{(MCLK)} = f_{(SMCLK)}$ = 4 096 Hz, $f_{(ACLK)}$ = 4,096 Hz XTS=0, SELM=(0,1) XTS=0, SELM=3	T_A = −40°C to 85°C	V_{CC} = 2.2 V		2.5	7	µA
			V_{CC} = 3 V		9	20	
$I_{(LPM0)}$	Low-power mode, (LPM0) (see Note 1)	T_A = −40°C to 85°C	V_{CC} = 2.2 V		32	45	µA
			V_{CC} = 3 V		55	70	
$I_{(LPM2)}$	Low-power mode, (LPM2), f(MCLK) = f (SMCLK) = 0 MHz, f(ACLK) = 32.768 Hz, SCG0 = 0	T_A = −40°C to 85°C	V_{CC} = 2.2 V		11	14	µA
			V_{CC} = 3 V		17	22	
$I_{(LPM3)}$	Low-power mode, (LPM3) $f_{(MCLK)} = f_{(SMCLK)}$ = 0 MHz, $f_{(ACLK)}$ = 32,768 Hz, SCG0 = 1 (see Note 2)	T_A = −40°C	V_{CC} = 2.2 V		0.8	1.5	µA
		T_A = 25°C			0.9	1.5	
		T_A = 85°C			1.6	2.8	
		T_A = −40°C	V_{CC} = 3 V		1.8	2.2	µA
		T_A = 25°C			1.6	1.9	
		T_A = 85°C			2.3	3.9	
$I_{(LPM4)}$	Low-power mode, (LPM4) $f_{(MCLK)}$ = 0 MHz, $f_{(SMCLK)}$ = 0 MHz, $f_{(ACLK)}$ = 0 Hz, SCG0 = 1	T_A = −40°C	V_{CC} = 2.2 V		0.1	0.5	µA
		T_A = 25°C			0.1	0.5	
		T_A = 85°C			0.8	2.5	
		T_A = −40°C	V_{CC} = 3 V		0.1	0.5	µA
		T_A = 25°C			0.1	0.5	
		T_A = 85°C			0.8	2.5	

NOTES: 1. Timer_B is clocked by f(DCOCLK) = 1 MHz. All inputs are tied to 0 V or to V_{CC}. Outputs do not source or sink any current.
2. Timer_B is clocked by f(ACLK) = 32,768 Hz. All inputs are tied to 0 V or to V_{CC}. Outputs do not source or sink any current. The current consumption in LPM2 and LPM3 are measured with ACLK selected.

Current consumption of active mode versus system frequency, F-version

I(AM) = I(AM) [1 MHz] × f(System) [MHz]

Figure 22: MSP430F149 Power Consumption

The screen copy of the MSP430F149 data sheet above shows, that at 3V and 1MHz, the MSP430F149 has a nominal active supply current of 420uA and a maximum of 560uA. As shown by the equation at the bottom of the page, the current is linearly proportional to frequency so the current at 8 MHz = 8 x the current at 1 MHz. The active current at 8

MHz is therefore 3.36mA nominal and 4.48mA maximum; an approximate value of 4mA was used for the comparison.

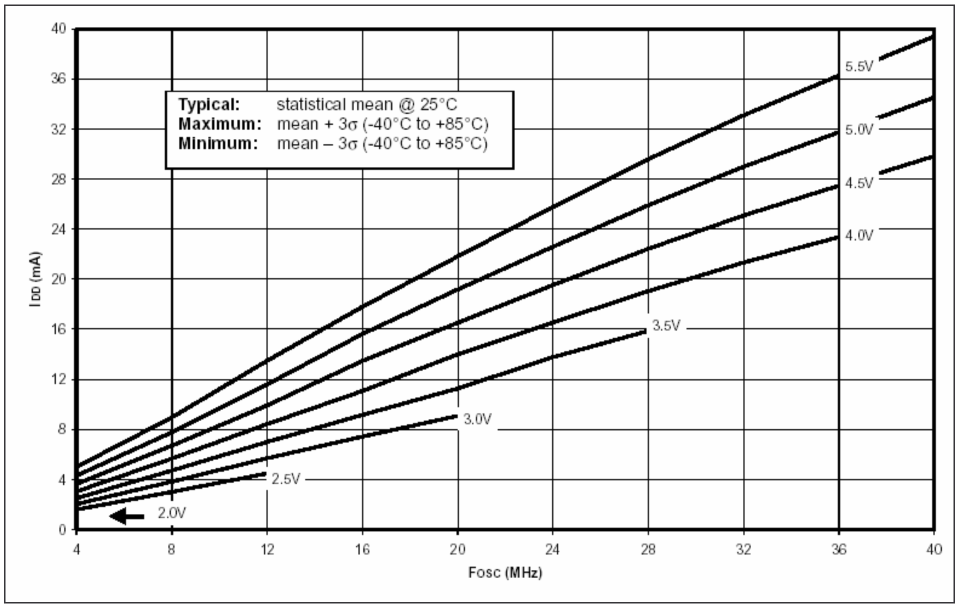


Figure 23: PIC18F6621 Typical Supply Current [From device datasheet.]

The PIC18F6621 power graphs show the supply current was about 4mA when operating at 8MHz and 3V.

The MSP430 and SBC65EC were both very good options. Neither product was clearly superior but the various comparisons gave the SBC65EC with the PIC18F6621 a very slight edge so it was chosen for the more detailed evaluation.

Note: The unit initially purchased (Dec 2005) came with the PIC18F6621 which has 64KB of Flash. Current versions of the SBC65EC now ship (as of May 2006) with the PIC18F6627 which has 96KB of Flash. All other parameters are essentially the same.

4. Evaluation of the Modtronix SBC65EC

4.1. Modtronix and Initial Impressions

One SBC65EC and two expansion (daughter) boards were ordered from Modtronix using their online ordering system. The parts arrived in about a week (Modtronix was located in Australia.) with no problems. The web site was well laid out and manuals, schematics and all source code was available online and easy to find. The manuals were clearly written and easy to follow. Overall, the organization and support provided by Modtronix gave a positive impression.

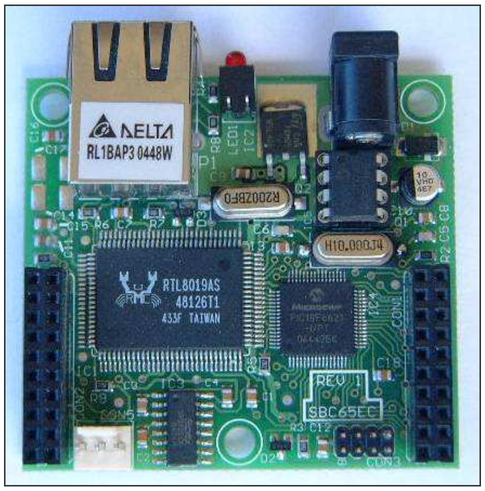


Figure 24: SBC65EC Top View (Approximately Full Size)

The SBC65EC board is shown in the above photograph. It was very well made, came properly packaged and worked correctly straight out of the box. The flashing red LED (center-top of photo) was a comfortable indication to a new user that the board was functioning.

The SBC65EC came loaded with the version 2.04 demo which included a web interface. All pages and functions available via the web interface were tested and no problems were found. The demo web pages provided a means of turning outputs on and off and reading the state of digital and analog inputs. These were all tested by connecting various analog and digital signals and everything worked correctly. Some of the web pages were a bit confusing but they were only provided as a demo of functionality not as a working application. Aside from a small issue with pings, which will be discussed next, everything worked fine and the initial impression was very positive.

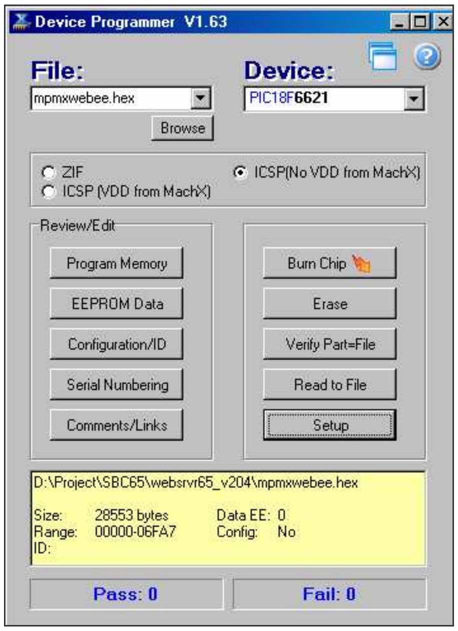
An unusual characteristic, that was not a serious problem but should be addressed in future upgrades, was discovered by accident during the very initial testing. After connecting power and Ethernet to the device it was pinged from a Cisco 2621 router on the same network. It did not reply. The first assumption, as the device was new, was that it was malfunctioning but it was not. Subsequent investigation revealed that it did not accept ping packets above a certain size. Cisco devices used a larger default ping size than DOS/Windows. The size values used by Cisco were different that DOS/Windows; Cisco used a "datagram" size of 100 and DOS/Windows used a "bytes" size of 32. Cisco's datagram size of 60 was equivalent to a DOS/Windows' "bytes" size of 32. The SBC65EC would only work with Cisco sizes of 36 to 60 which was equivalent to DOS/Windows' bytes sizes of 0 to 32.

4.2. Programming the SBC65EC

4.2.1. ICSP

The compiler, which will be discussed shortly, produced executable binary code that was packaged into a format for transmission. This packaged code was displayed in hexadecimal format and had a ".hex" file extension so it was generally referred to as "hex code".

The SBC65EC used a PIC18F6621 uC and the executable code was stored in the on-chip flash. An in-circuit serial programmer (ICSP) was required to transfer the hex code to the chip and the SBC65EC had a connector (Figure 24, lower-right corner) specifically for this purpose. To evaluate the easy of making software changes and reprogramming the device a small inexpensive ($10) ICSP with a USB interface was purchased. It was a poor decision as it was difficult to use and there were many problems with the USB drivers for it. A better quality programmer, the MACH-X, was purchased from Custom Computing Services (CCS) for $199 US. Installation and use were easy and the user interface was simple and intuitive. A screen copy of the user interface is shown below.



File: mpmxwebee.hex
Device: PIC18F6621

ZIF
ICSP (VDD from MachX)
[✓ ICSP(No VDD from MachX)]

D:\Project\SBC65\websrvr65_v204\mpmxwebee.hex
Size: 28553 bytes   Data EE: 0
Range: 00000-06FA7   Config: No
ID:

Pass: 0   Fail: 0

Figure 25: MACH-X User Interface

The MACH-X had an RJ11 output connector and the SBC65EC had an 8-pin 2mm male header. Both Modtronix and CCS provided pinout information so adaptor cables could be fabricated. The programmer, adaptor cable, and SBC65EC are shown in the photos below. All schematics are included for reference in Appendix C.

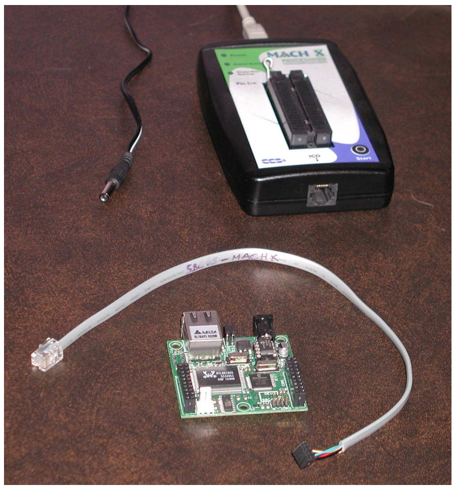


Figure 26: MACH-X Programmer and ICSP Adaptor Cable

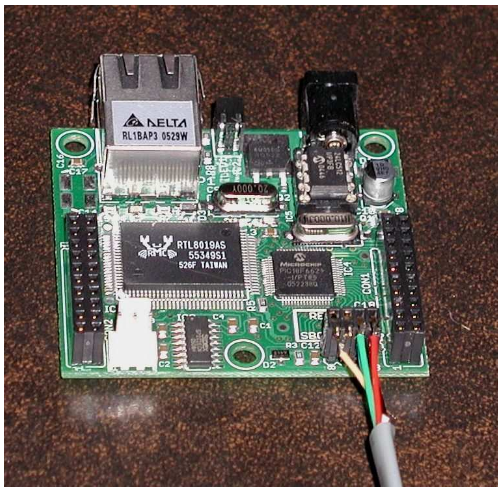


Figure 27: ICSP Adaptor Cable and SBC65EC Close-up

According to the PIC18F6621 data sheet the flash can typically be reprogrammed 100,000 times.

4.2.2. Integrated Development Environment

The Integrated Development Environment (IDE) from Microchip was called MPLAB. It was free, well supported and very complete. Version 7.30, a 34MB zip file, was downloaded from the Microchip site. Installation was simple and took only a few minutes. The IDE was very complete and had many advanced features for debugging and simulation. Because of the power and complexity of the IDE it did take a couple of days to get comfortable with the environment, but the documentation was clear and well written and there were several online tutorials to help get over the initial learning curve. It took about three days in total to set up and get familiar with the IDE, load the demo project for the SBC65EC, compile the code and reprogram the uC (using the ICSP previously discussed). Subsequent program changes were very straight forward. The overall impression of MPLAB, Microchip's IDE, was very positive.

4.2.3. C Compiler

There were several free and commercial C compilers available for the PIC18 uC's. The demo program for the SBC65EC was written for both Microchip's own compiler (C18) and the Hi-Tech compiler. Both compliers retailed for around $1,000US but the Microchip C18 compiler was also available in a free student version so that seemed the best option.

The MPLAB® C18 compiler is a full-featured ANSI compliant C compiler for the PIC18 family of PICmicro® 8-bit MCUs. MPLAB C18 is a 32-bit Windows® console application as well as a fully integrated component of Microchip's MPLAB Integrated Development Environment (IDE), allowing source level debugging with the MPLAB ICE In-Circuit Emulator, the MPLAB ICD 2 In-Circuit Debugger, and the MPLAB SIM simulator.

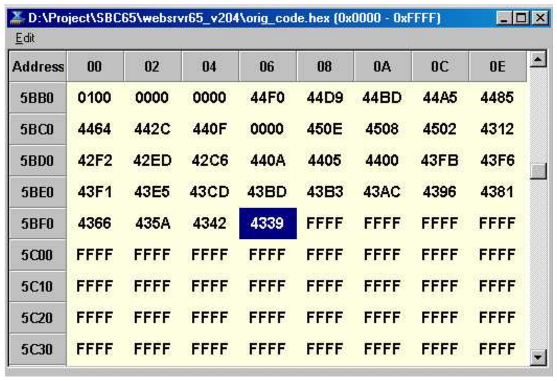
…

Student Edition/Demo

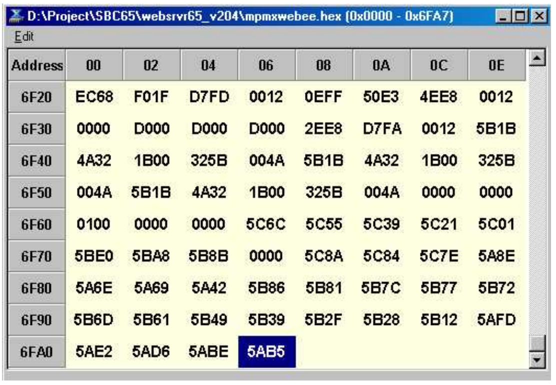
The Student Edition is free! It has all the features of the full compiler and libraries. After 60 days, the optimizations related to procedural abstraction and to the extended instruction set of the newer PIC18XXXX devices will be disabled. Code compiled after the expiration date will function but may occupy more memory space. [17]

The full-featured trial version was downloaded and installed with no problems. The full-featured version was used for the initial tests with the IDE then the trial period was allowed to elapse and the student version was used. Both compiled with no problems. The hex code generated by each compiler was used to reprogram the PIC18F6621 uC and everything worked exactly the same. As stated by Microchip, the only difference was the size of the executable code.

The file size could not be used directly to determine to size of the executable code. The hex code could be viewed directly but it was difficult to decipher. The MACH-X had a "display program memory" option to see the executable code and it was used to generate the two screen captures shown below. The first code listing, generated by the full-feature commercial (60 day trial) version, ended at 0x5BF7 = 23,543 and the second code listing generated by the student version ended at 0x6FA7 = 28,583. The difference was 5,040 bytes or about 25%. If smaller code size is needed the commercial version can be purchased otherwise the student version is fine.


D:\Project\SBC65\websrvr65_v204\orig_code.hex (0x0000 - 0xFFFF)
Edit

Address	00	02	04	06	08	0A	0C	0E
5BB0	0100	0000	0000	44F0	44D9	44BD	44A5	4485
5BC0	4464	442C	440F	0000	450E	4508	4502	4312
5BD0	42F2	42ED	42C6	440A	4405	4400	43FB	43F6
5BE0	43F1	43E5	43CD	43BD	43B3	43AC	4396	4381
5BF0	4366	435A	4342	4339	FFFF	FFFF	FFFF	FFFF
5C00	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
5C10	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
5C20	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF
5C30	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF	FFFF


D:\Project\SBC65\websrvr65_v204\mpmxwebee.hex (0x0000 - 0x6FA7)
Edit

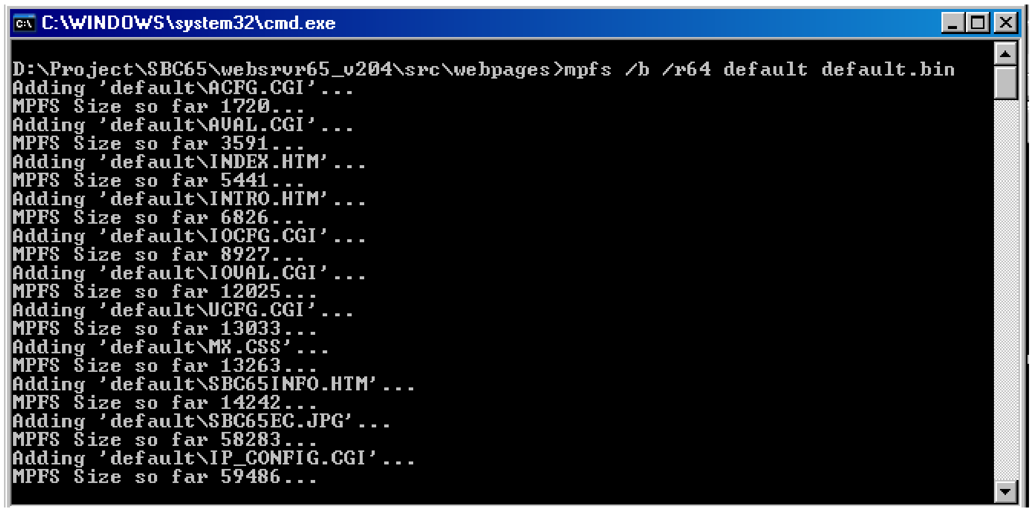
Address	00	02	04	06	08	0A	0C	0E
6F20	EC68	F01F	D7FD	0012	0EFF	50E3	4EE8	0012
6F30	0000	D000	D000	D000	2EE8	D7FA	0012	5B1B
6F40	4A32	1B00	325B	004A	5B1B	4A32	1B00	325B
6F50	004A	5B1B	4A32	1B00	325B	004A	0000	0000
6F60	0100	0000	0000	5C6C	5C55	5C39	5C21	5C01
6F70	5BE0	5BA8	5B8B	0000	5C8A	5C84	5C7E	5A8E
6F80	5A6E	5A69	5A42	5B86	5B81	5B7C	5B77	5B72
6F90	5B6D	5B61	5B49	5B39	5B2F	5B28	5B12	5AFD
6FA0	5AE2	5AD6	5ABE	5AB5				

Figure 28: C18 Compiler Code, Full-Feature Left, Student Version Right

4.2.4. Web Pages

The SBC65EC stored web pages in the external EEPROM. The device came with a 64KB EEPROM (24LC512) chip in a socket so it could be replaced with a larger device if more space was needed. The Modtronix demo used about half of the available space.

To optimize the use of limited memory Microchip had developed a special file system (MPFS) and a compression method for standard HTML web pages. Details are available on page 83 of AN833 [17]. The Modtronix manuals gave clear instructions on how to compress web pages and they provided a batch file to ease the process. A sample output from that batch file is shown below.



```
D:\Project\SBC65\websrvr65_v204\src\webpages>mpfs /b /r64 default default.bin
Adding 'default\ACFG.CGI'...
MPFS Size so far 1720...
Adding 'default\AVAL.CGI'...
MPFS Size so far 3591...
Adding 'default\INDEX.HTM'...
MPFS Size so far 5441...
Adding 'default\INTRO.HTM'...
MPFS Size so far 6826...
Adding 'default\IOCFG.CGI'...
MPFS Size so far 8927...
Adding 'default\IOVAL.CGI'...
MPFS Size so far 12025...
Adding 'default\UCFG.CGI'...
MPFS Size so far 13033...
Adding 'default\MX.CSS'...
MPFS Size so far 13263...
Adding 'default\SBC65INFO.HTM'...
MPFS Size so far 14242...
Adding 'default\SBC65EC.JPG'...
MPFS Size so far 58283...
Adding 'default\IP_CONFIG.CGI'...
MPFS Size so far 59486...
```

Figure 29: Sample "doall.bat" Output

Compressed web pages were uploaded to the SBC65EC by FTP. WS_FTP LE (Limited Edition) was used with some minor problems. The FTP code on the SBC65EC was a very minimal implementation to save space so it did not support all functions. It generally took two to three attempts to transfer the file. When the transfer failed it would hang and take several minutes time out. Sometimes power cycling the SBC65EC was the quickest way to clear the problem. Version 2.04 was used for this evaluation and current products were shipping with version 3.04 so some of these problems may have already been fixed.

To evaluate the ease or difficulty of creating and editing web pages, new pages were created for this project. The manual for creating and modifying web pages had very clear and easy to follow instructions but modifications of existing pages and creation of new pages was done with a text editor directly in HTML to keep the pages small and simple. The new web pages were not complex but coding directly in HTML was definitely tedious. The demo web pages had a lot of functionality but all html functions were not supported. As stated in the manual, file names could not have any of the following non-alphanumeric characters or pages would become inaccessible (lock up) with no warning:

- single or double quotes (‘ and “)
- left or right angle brackets (< and >)
- pound sign (#)

- percent sign (%)
- left or right brackets or braces ([,{,] and })
- pipe symbol (|)
- backslash (\ )
- caret (^)
- tilde (~)

Only *.htm, *.cgi and *.jpg files were used in the modified pages but the manual stated that the server also supported: *.txt, *.gif, *.cla, and *.wav files.

The "SBC65EC" page includes a 43KB photo of the SBC65EC and was configured to automatically reload every 3 seconds so it could be used for testing.

4.3. Expansion Board

Subsequent development of a sensor net node would be facilitated by simple expansion capability. In particular it would likely be necessary to add circuits like an RTC, optical communications, and possibly special interfaces for environmental sensors.

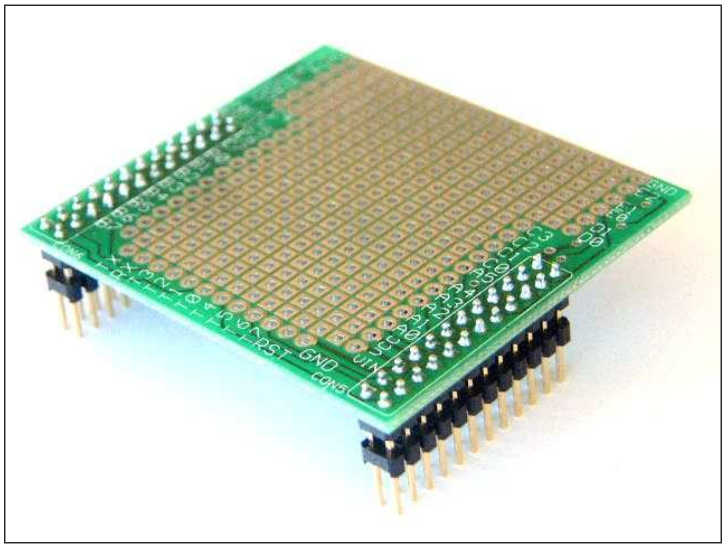


Figure 30: Unmodified SBC65EC Expansion Board

An expansion board was modified by adding a simple environmental sensor and circuits to interconnect various I/O lines. The four separate circuits that were constructed were a:

1. light sensor that is enabled by output C0 and is measured by analog input A5
2. R2R ladder network that connects outputs B0, B1, B2 and B3 to analog input A0
3. circuit that connects output C1 to analog inputs A1, A2 & A3
4. circuit that connects output C2 to analog inputs F0 thru F6

The schematic is included for reference in Appendix C.

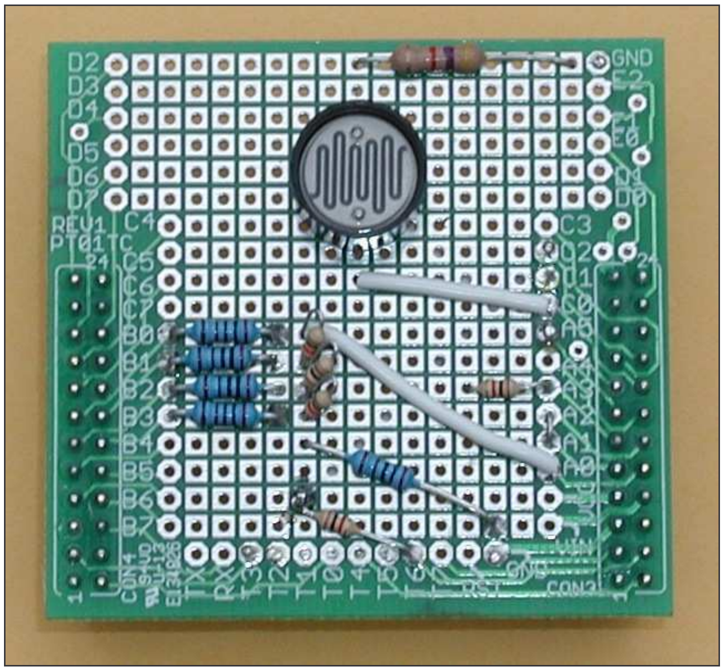


Figure 31: Modified SBC65EC Expansion Board

The expansion board was well made, all of the I/O power and ground lines were available and most* were clearly labeled.

The expansion connectors had 12 pairs of pins but the mating connector on the SBC65EC had only 10 pairs of pins as can be seen in the photo below. It was a bit confusing at first but pin 1 was labeled on both boards and when mated the two boards align so it was quickly resolved.

The labeling errors and connector size differences implied this expansion board was intended for use with more than one SBC.

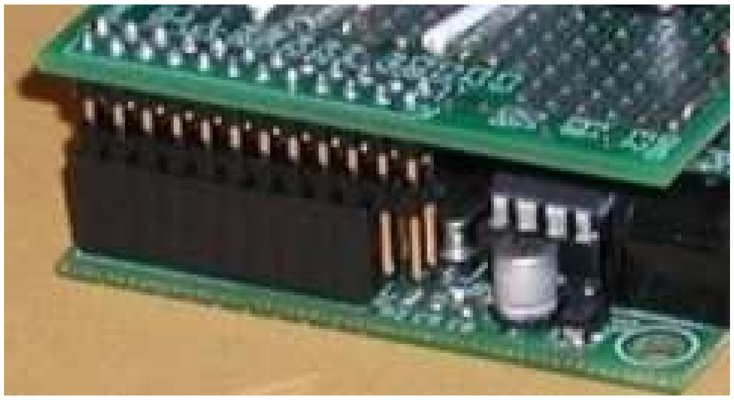


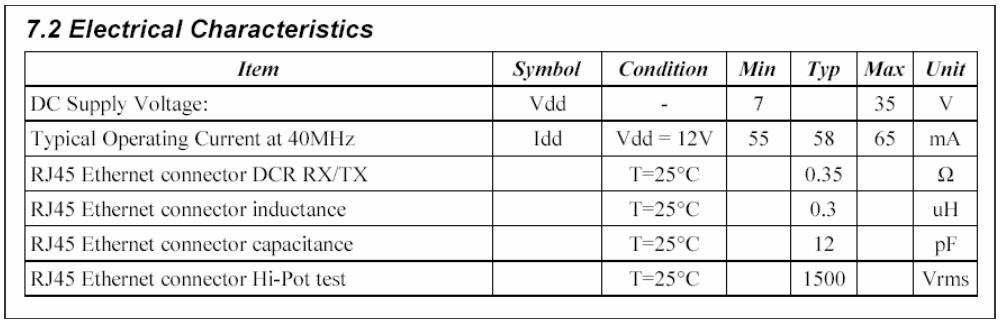
Figure 32: Expansion Board Connector

* The labels for the "T" connection points across the bottom actually connected to port "F" pins.

4.4. Power Consumption

Power consumption is a concern for any battery-operated device, a major concern for sensor nets in general, and a critical issue for an underwater sensor net.

The power requirements specified in the SBC65EC manual are shown below. Because all of the SBC's use on-board voltage regulators the supply voltages were not used to calculate the power ratings of the boards, as stated earlier. Like most of the other boards the SBC65EC lists a supply voltage of 12V but it works with a range of supply voltages. System power was calculated using the typical current of 58mA at the operating voltage of the board which in this case is 5V, which gives 290mW.



7.2 Electrical Characteristics

Item	Symbol	Condition	Min	Typ	Max	Unit
DC Supply Voltage:	Vdd	-	7		35	V
Typical Operating Current at 40MHz	Idd	Vdd = 12V	55	58	65	mA
RJ45 Ethernet connector DCR RX/TX		T=25°C		0.35		Ω
RJ45 Ethernet connector inductance		T=25°C		0.3		uH
RJ45 Ethernet connector capacitance		T=25°C		12		pF
RJ45 Ethernet connector Hi-Pot test		T=25°C		1500		Vrms

Figure 33: SBC65EC Power Consumption

The actual power consumption of the SBC65EC was measured under three different situations:

- With a 40MHz clock, 5V supply
- With a 10MHz clock and 5V supply
- With a reduced supply voltage

For the 40MHz and 10MHz clock speed measurements, the supply current was measured with the device idling and when a large file was being transferred. The "SBC65EC" page on the modified web site contained a large (43KB) image of the SBC65EC board specifically for the test. The results are summarized in the table below.

A test with the Ethernet controller disabled was investigated but it was not practicable to perform.

Table 4: Summary of Average Supply Current Tests

Test Condition	Normal Operation	Large Download
40MHz, 5V	54.3mA	58.1mA
10MHz, 5V	36.6mA	39.3mA
40MHz, 2.5V	25mA	N.A.

4.4.1. Test Setup

A block diagram and photograph of the test setup are shown below.

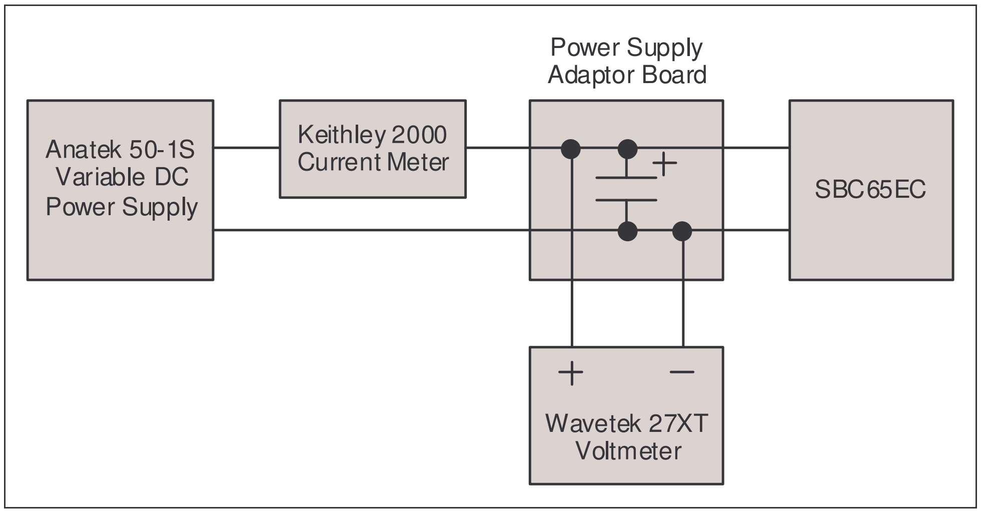

Anatek 50-1S Variable DC Power Supply
Keithley 2000 Current Meter
Power Supply Adaptor Board
SBC65EC
Wavetek 27XT Voltmeter
+
−

Figure 34: Test Setup, Block Diagram

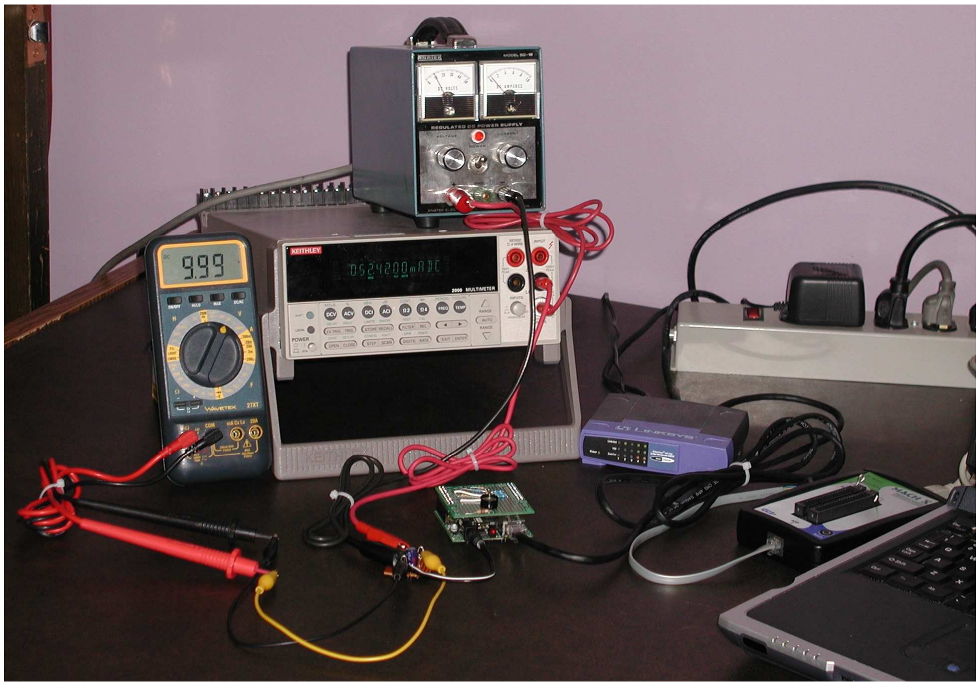


Figure 35: Test Setup, Photograph

The above photograph of the test setup shows the Wavetek 27XT voltmeter on the left, the Keithley 2000 current meter center-back, the Anatek 50-1S variable power supply on top of the Keithley 2000, and the SBC65EC with the power supply adaptor board in the front. The MACH-X programmer on the left was not connected during testing.

The small power supply adaptor board, shown connected to the SBC65EC with the modified expansion board, was fabricated to provide a convenient connection point for the voltage and current meters. A 470uF, 15V electrolytic capacitor was used to help reduce noise and decouple any inductive effects from the cables to the adaptor board.

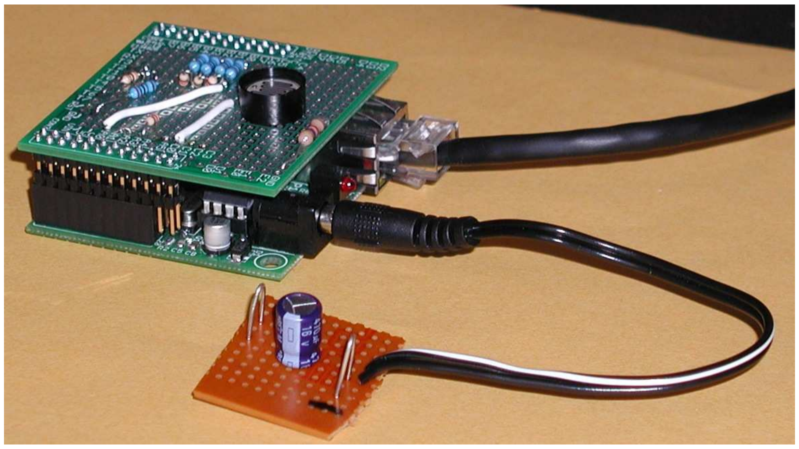


Figure 36: Power Supply Adaptor Board

For all tests the Kiethley 2000 was set to 100mA (fixed) current range, medium sampling rate, and to average 100 readings. With these settings the display updated approximately every 2 seconds.

With either supply lead disconnected the current meter read at or less than 0.0084mA. To determine the combined effects of the adaptor board and DVM on the test results (due to factors such as capacitor leakage and DVM loading) tests were done at 12V, 10V and 8V. For each test the DVM and SBC65EC were connected and running for at least 5 minutes to ensure stable operation, then the SBC65EC was unplugged from the power supply and the current monitored for the next 5 minutes. At all three voltages the reading was at or below 0.0097mA. An offset error of less than 0.01mA was acceptable as the tests results were recorded to 0.1mA.

The Keithley 2000 must be set to the fixed 100mA range to determine offset errors. When allowed to auto-range (which it does by default) it changed from the 100mA range when reading the test currents to the 10mA range when measuring offset. The offset on the 10mA range was below 0.0001mA; 2 orders of magnitude better than the range used to make the test measurements.

Some current values fluctuated noticeably over a 5 to 10 second period. If this occurred the test result was calculated by recording 10 successive current readings, which took about 20 seconds, then averaging the results.

4.4.2. 40MHz Clock Speed

All of the these tests were done with:

- The modified daughter board connected
- 10V supply voltage
- Browser accessing Intro page (if Ethernet connected)

- I/O pins B0-3, B6, C0-2 set to output with B0-3 set low, C0-2 set high
- Heartbeat enabled (because B6 is set to output)
- A/D converter on for all 12 channels
- Clock frequency of 40MHz

Normal operating current was approximately 54mA and it increased to 58mA when performing large file transfers.

The heartbeat LED and the Ethernet connector both account for small but measurable amounts of total current. The SBC65EC has a red "heartbeat" LED; that flashes, when enabled, to indicate the board is working. The LED is connected to pin B6 so it can be easily disabled by setting the I/O direction on pin B6 to "input". The average current changed from 53.6mA when disabled, to 54.3mA when enabled.

The Ethernet connector has two LED's, a green link-status and a yellow activity indicator. The total average current was 53.6mA with the Ethernet connected and 52.1mA when disconnected. This test was done with the Ethernet connected and working then disconnected, and when the board was power-on with no Ethernet connected; the result was the same for both cases.

No measurable changes occurred when the A/D converter was on or off.

Only output lines B0-3, B6, and C0-2 were connected to external circuits. When all of these lines were turned off (with the exception of B6 which controlled the heartbeat LED) the average total current was 52.7mA when all lines were on the current was 55.1mA.

The most significant change in normal operating current occurred when the SBC65EC was transferring large files; it increased from 54.3mA to 58.1mA. This test was done by continuously refreshing the "SBC65EC" page with the large image that was added to the server for this test.

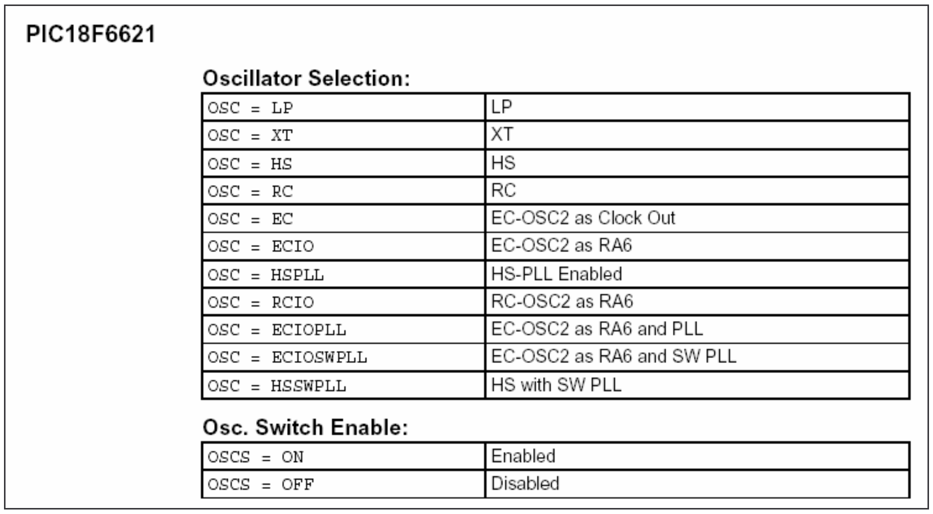
4.4.3. 10MHz Clock Speed

The SBC65EC used a 10MHz resonator and the phase-lock loop (PLL) option on the PIC18F6621 to get a clock speed of 40MHz.

The PIC18F6621 datasheet describes the options for the PLL:

> *The PLL can only be enabled when the oscillator configuration bits are programmed for High-Speed Oscillator or External Clock mode. If they are programmed for any other mode, the PLL is not enabled and the system clock will come directly from OSC1. There are two types of PLL modes: Software Controlled PLL and Configuration Bits Controlled PLL. In Software Controlled PLL mode, PIC18F6525/6621/8525/8621 executes at regular clock frequency after all Reset conditions. During execution, the application can enable PLL and switch to 4x clock frequency operation by setting the PLLEN bit in the OSCCON register. In Configuration Bits Controlled PLL, the PLL operation cannot be changed "on-the-fly".*

The Microchip programming manual for the C18 compiler specified the commands to set the different oscillator modes for the PIC18F6621.



PIC18F6621

Oscillator Selection:

`OSC = LP`	LP
`OSC = XT`	XT
`OSC = HS`	HS
`OSC = RC`	RC
`OSC = EC`	EC-OSC2 as Clock Out
`OSC = ECIO`	EC-OSC2 as RA6
`OSC = HSPLL`	HS-PLL Enabled
`OSC = RCIO`	RC-OSC2 as RA6
`OSC = ECIOPLL`	EC-OSC2 as RA6 and PLL
`OSC = ECIOSWPLL`	EC-OSC2 as RA6 and SW PLL
`OSC = HSSWPLL`	HS with SW PLL

Osc. Switch Enable:

`OSCS = ON`	Enabled
`OSCS = OFF`	Disabled

Figure 37: PIC18F6621 Oscillator Options [from C18 compiler manual]

The source code changes required for operation at 10MHz were complex and required changes to five different sections. The USART code was hard-coded for a 40MHz clock and there was no method of disabling it even though it was not being used. Numerous changes were needed in the delay routines as they were not configured for operation at different clock speeds.

The original and modified code sections are included for reference in Appendix D.

Because the version 2.04 code supplied with the SBC65EC was a modified version of the original Microchip code it was not clear if the poorly structured code was due to Microchip or Modtronix. The overall impression was that most of the source code was very well structured and well commented but some sections and/or modifications of those sections (the USART and delay portions in particular) had been written very poorly.

On the plus side it provided an in-depth and very positive exposure to the compiler and IDE. The C18 compiler provided good diagnostic warning and error messages and the MPLAB IDE made it easy to manage changes.

The same tests that were done for normal operation were repeated with the uC clock running at 10MHz. Everything worked the same just a bit slower; the refresh speed of the "SBC65EC" web page was the most noticeable.

The 10MHz tests were done with:

- The modified daughter board connected
- 10V supply voltage
- Browser accessing Intro page (if Ethernet connected)
- I/O pins B0-3, B6, C0-2 set to output with B0-3 set low, C0-2 set high
- Heartbeat enabled
- A/D converter on for all 12 channels
- Clock frequency of 10MHz

The average current with Ethernet connected and heartbeat enabled was:

- 54.3mA at 40MHz
- 36.6mA at 10MHz

When transferring large files the average current was:

- 58.1mA at 40MHz
- 39.3mA at 10MHz

The current vs. clockspeed curves for the PIC18F6621 [Figure 23] showed currents of approximately 32mA at 40MHz and 12mA at 10MHz (a change of 20mA) which agreed very well with the measured values.

4.4.4. Supply Voltage Reduction

The voltage on the SBC65EC (Vcc) is kept at a constant 5V by the onboard 78M05 voltage regulator as long as the supply voltage is at or above 7.25V. The current stays constant at supply voltages above 7.5V but drops linearly with voltage as shown by the graph below. Both the supply voltage (green, circles) and the SBC65EC Vcc voltage (blue, squares) are shown overlaid with lines (red) indicating when the heartbeat (dashed) was active and when the Ethernet access (solid) was working.

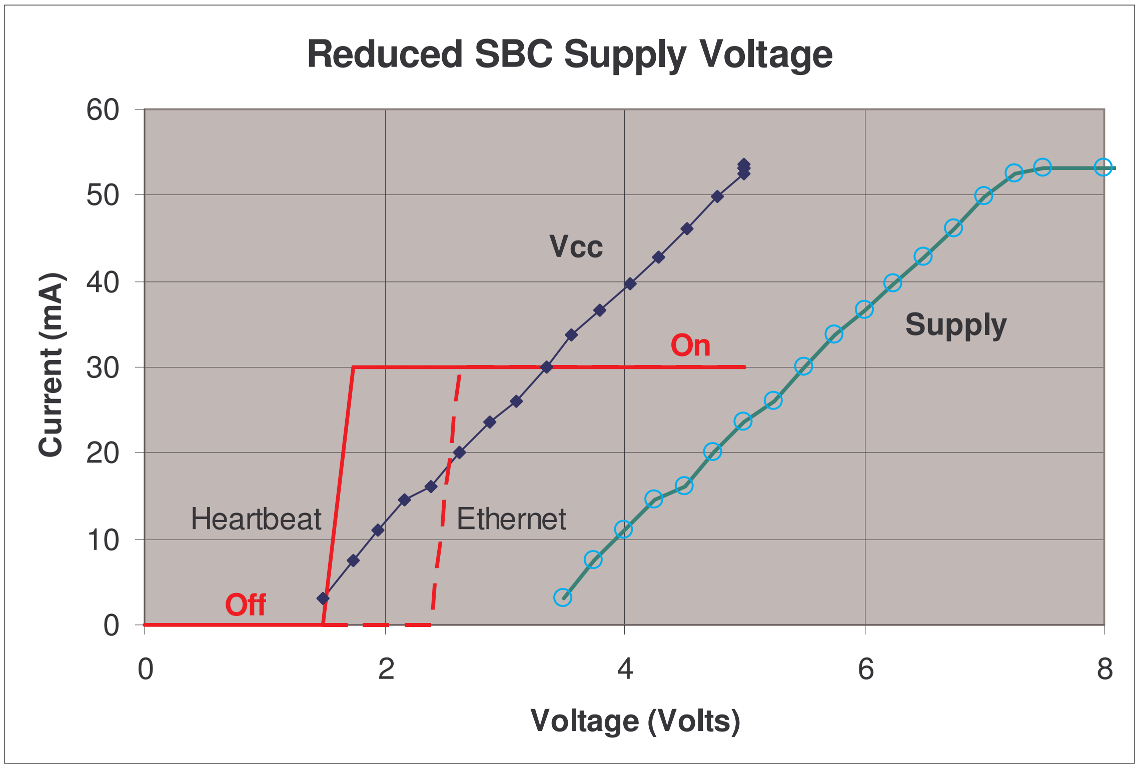


Figure 38: Reduced SBC Supply Voltage

The specification sheet for the Realtek RTL8019AS Ethernet controller guarantees operation when Vcc is 5.0V +/- 5%. The device worked to well below 3V. The

PIC18F6621 is guaranteed to work to 2V and was operating almost all the way down to 1.5V.

4.4.5. Ethernet Controller

The data sheet for the Realtek RTL8019AS Ethernet controller stated that the device supports three power-down modes:

1. Sleep
2. Power down with internal clock running
3. Power down with internal clock halted

The procedure specified in the data sheet to implement the power-down was straight forward:

> *To set the device to lowest power consumption mode the clock must be halted:*
>
> - *Set COMMAND register (01h) to 11xxxxxx, to allow config registers to be written.*
> - *Set HLTCLK register (09h) to 01001000 (ascii letter "H") to set mode to "clock halted"*
> - *Set CONFIG3 register (06h) to 11xxxxxx, to power-down device.*
>
> *In this mode only CONFIG3 can be written to power device back on.*

A detailed analysis of the version 2.04 source code indicated that many different sections of code were dependant on responses from the Ethernet controller so code modifications to power-down the RTL8019AS chip were not practicable.

5. Conclusions

The market research and the subsequent evaluation of the SBC65EC support the following conclusions:

5.1. There were many very good SBC's on the market. Rabbit Semiconductor, Digi and Netburner all had excellent products. If time was a major concern one of these would have been the best option. The Rabbit Semiconductor RCM3750 SBC, had lots of memory, a real-time OS, and included a real-time clock (RTC). All of them however were proprietary in terms of hardware and software, so they were not good options for long-term collaborative development.

5.2. The most suitable products were based on two uC families; the Texas Instruments MSP430 series and the Microchip PIC18 series. Both had very low power requirements, similar features and memory, good development support, and open designs. A sensor net node could be developed around either of these two product families.

5.3. The Modtronix SBC65EC would be a good platform to use for development of an underwater sensor net node. The product was well made, readily available, and reasonably priced. Modtronix had a well-organized web site with all documents and code easily accessible and freely available. The software and physical design were both open. The Microchip PIC18F6621 used on the SBC65EC had excellent support and development tools available from Microchip. The Microchip C18 compiler was an ANSI-standard C-compiler, with both free and commercial versions available. Microchip was continuing development of newer products in the PIC18 family with more capacity that were code and hardware compatible with existing products. (The SBC65EC had already migrated to the PIC18F6627 with more memory.) Both open (FreeRTOS) and proprietary (Salvo) OS's were available for this platform.

5.4. The actual power consumption of the SBC65EC was slightly better than the manufacturer's specification. Software changes such as reducing the clock speed, and putting the uC and Ethernet controller in standby when not used, could reduce power consumption but the SBC65EC cannot operate at ultra-low power. The uC needs only a few uA's sleeping and the EEPROM even less (0.1uA standby) but the RS232 chip (MAX202E) requires 8mA of operating current (15mA maximum) and does not have a low power mode. Low voltage operation can significantly reduce power for the uC but it is not a viable option with this SBC; the uC and EEPROM chips are specified to operate at low voltage, but the Ethernet controller and RS232 chips are not.

6. Recommendations

The following recommendations are offered for further research and development:

6.1. Additional research is needed to evaluate real-time operating systems for use in very-low power applications in general and sensor nets in particular. An evaluation of TinyOS, FreeRTOS, and Salvo should be done before any decision is made to use or not use an OS for the sensor net node development.

6.2. Initial development can be done using the SBC65EC, but rather than design an expansion board to work with the SBC65EC a new board should be designed that includes everything on one board. The SBC65EC design is both simple and open. If a board needs to be designed and built for the optical-layer circuits and RTC anyway, rather than design it as an expansion board, it would be simpler to just build one board that includes all of the components and circuits of the SBC65EC and the expansion.

Designing a new board would also allow component changes for ultra low-power operation. Lower voltage was shown to significantly reduce power but not all devices are rated to work at lower voltage. The PIC18F6627 operates from 2.0 to 5.5V and the 24LC512 (EEPROM) operates from 2.5 to 5.5V, but the MAX202E serial interface chip, on the SBC65EC, that does not operate below 5V. If RS-232 communication is not needed it could be removed entirely, but if it is needed it could be replaced by a device like the MAX3218 which operates down to 1.8V and includes a no activity auto shutdown that drops the supply current to 10uA.

The Realtek RTL8019AS Ethernet controller is not rated for operation below 5V[*] but it is not needed for nodes with no Ethernet connection. The system could detect the presence or absence of Ethernet, and if present run at 5V and if not present place the Ethernet controller in sleep mode then drop the system voltage to 2.5V.

6.3. The optical layer should be implemented using a second uC as the optical controller. Keeping the communications controllers separate from the main processing maintains the architecture already used for the Ethernet controller and will facilitate development.

[*] The RTL8019AS did actually work properly at lower voltages so it may be just an oversight in the datasheet.

7. References

[1] Cui J., Kong J., Gerla M, Zhou S. , “Challenges: Building Scalable and Distributed Underwater Wireless Sensor Networks (UWSNs) for Aquatic Applications, UbiNet-TR05-02, 2005, http://www.cse.uconn.edu/~jcui/UWSN_papers/UCONN_CSE_UbiNet-TR05-02.pdf

[2] Culler D, Hill J., Buonadonna P., Szewczyk R., Woo A. “A Network-Centric Approach to Embedded Software for Tiny Devices”, 2001, http://www.cs.berkeley.edu/~awoo/embedded.pdf

[3] Custom Computing Services (CCS), http://www.ccsinfo.com/

[4] Digi, http://www.digi.com/

[5] Dunkels A., “uIP“, http://www.sics.se/~adam/uip/

[6] Dunkels A, Feeney L., Grönvall B., Voigt T., “An integrated approach to developing sensor network solutions”, 2004, http://www.sics.se/~adam/sanpa2004.pdf

[7] Dunkels A., Grönvall B., Johansson M., Mayer K., Oldewurtel F., Raivio O., Riihijärvi J., “Existing Architectures and Components”, RUNES D1.1, 2004, http://www.ist-runes.org/public_deliverables.html,

[8] Ethernut, http://www.ethernut.de/en/index.html

[9] Free RTOS, http://www.freertos.org/

[10] Gay D., Levis P., Von Behren R., Welsh M., Brewer E., Culler D., “The nesC Language: A Holistic Approach to Networked Embedded Systems”, 2003, http://www.eecs.harvard.edu/~mdw/papers/nesc-pldi-2003.pdf

[11] Johansson M., Faulkner M., Oldewurtel F., Kannegiesser P., “Review of Existing Technologies”, RUNES D3.2, 2005, http://www.ist-runes.org/public_deliverables.html,

[12] Jurdak R., Baldi P.,Lopes C., “State-Driven Energy Optimization in Wireless Sensor Networks”, 2005, http://www.isr.uci.edu/~lopes/documents/senet%2005/senet05.pdf

[13] Lee E. “Embedded Software”, Advances in Computers Vol. 56, Academic Press, London, 2002, http://ptolemy.eecs.berkeley.edu/publications/papers/02/embsoft/embsoftwre.pdf

[14] Levis P., “TinyOS Programming”, 2006, http://csl.stanford.edu/~pal/pubs/tinyos-programming-1-0.pdf

[15] Linux Devices, http://www.linuxdevices.com/news/NS5044199820.html

[16] Micro/Sys, http://www.embeddedsys.com/

[17] Microchip, http://www.microchip.com

[18] Modtronix, http://www.modtronix.com/

[19] Netburner, http://www.netburner.com/

[20] NOAA, ARGOS, http://noaasis.noaa.gov/ARGOS/

[21] Olimex, http://www.olimex.com/

[22] PICNIC, http://picnic.sourceforge.net/

[23] Rabbit Semiconductor, http://www.rabbitsemiconductor.com/

[24] SoftBaugh, http://www.softbaugh.com/

[25] Technologic Systems, http://www.embeddedarm.com/

[26] VENUS Documents, http://www.venus.uvic.ca/documents/index.html

[27] VENUS Home, http://www.venus.uvic.ca/index.html

[28] TinyOS, http://www.tinyos.net/

[29] Turley J. “Embedded systems survey; Operating systems up for grabs”, 2005, http://www.embedded.com/showArticle.jhtml?articleID=163700590

[30] Stevenson R. “The Future of Embedded Systems”, SEA 2002, http://www.cyberguard.com/download/white_paper/en_cg_embedded_systems.pdf

[31] Wikipedia, http://en.wikipedia.org/

Appendix A. Single Board Computers Surveyed

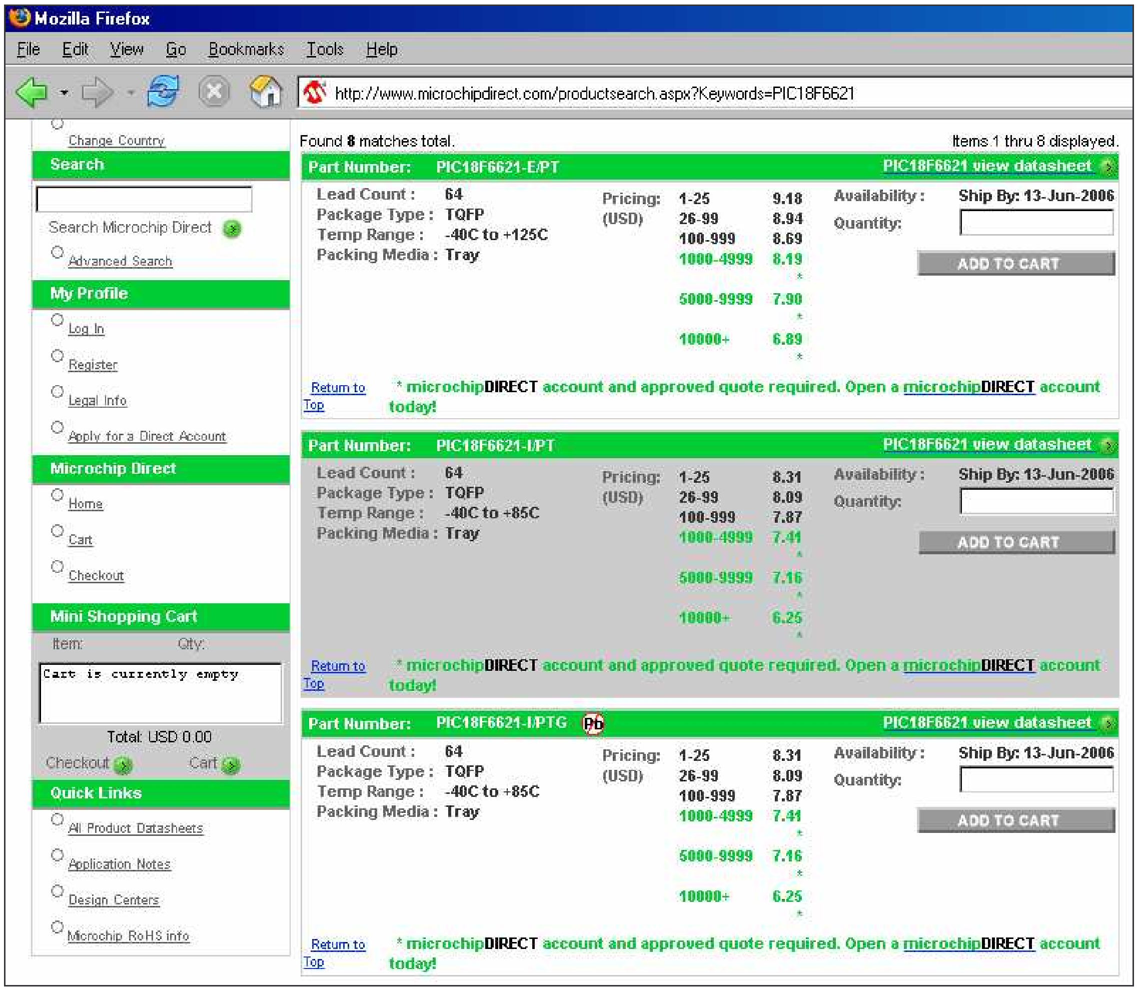
- Crossbow
  - TelosB
  - Mica2
  - MicaZ
  - Cricket
- Digi
  - ConnectCore XP
  - ConnectCore 9P
  - ConnectCore 9C
  - ConnectCore 9U
  - ConnectCore 7U
  - ConnectCore Wi-9C
  - Digi Connect ME
  - **Digi Connect EM, (Included in comparison.)**
  - Digi Connect Wi-ME
  - Digi Connect Wi-EM
- EDTP Electronics Inc.
  - Easy Ethernet /W
  - Easy Ethernet /AVR
  - Easy Ethernet /CS8900
  - Easy Ethernet /FT
- Ethernut
  - Ethernut 1.3
  - **Ethernut 2.1 (Included in comparison.)**
  - Ethernut 3.0
- HWGroup
  - Charon 1
  - EZL-50
  - EZL-60
- IPSil
  - IPu8930
  - IPu8932
- Lantronix
  - XPort
  - XPortAR
  - Micro
  - Micro 100
  - Mini
  - UDS10B
  - UDS100B
- Moteiv
  - Tmote Sky

- Micro/Sys
  - SBC1670
  - SBC1625
  - SBC1586
  - SBC1495
  - **SBC1491 (Included in comparison.)**
  - SBC1486
  - SBC1390
  - SBC1386
  - SBC1190
  - SBC1188
- Modtronix
  - SBC44B
  - SBC44EC
  - SBC45EC
  - **SBC65EC (Included in comparison.)**
  - SBC68EC
- NetBurner
  - MOD5213
  - **MOD5270 (Included in comparison.)**
  - MOD5272
  - MOD5282
- Olimex, ARM line
  - LPC-H2103
  - LPC-H2106
  - LPC-H2124
  - LPC-H2129
  - LPC-H2138
  - LPC-H2148
  - LPC-H2214
  - LPC-H2294
  - LPC-P2103
  - LPC-P2106
  - LPC-P2124
  - LPC-P2129
  - LPC-P2138
  - LPC-P2148
  - LPC-MT-2106
  - LPC-MT-2138
  - **LPC-E2124 (Included in comparison.)**
  - LPC-E2129
  - LPC-E2214
  - LPC-E2294
- Olimex, MSP430 Line
  - MSP430-169LCD
  - MSP430-1121STK2
  - MSP430-169STK
  - MSP430-413STK2
  - MSP430-417STK2
  - MSP430-449STK2
  - MSP430-EasyWeb2
  - **MSP430-EasyWeb3 (Included in comparison.)**

- Rabbit Semiconductor
    - RCM4100 RabbitCore
    - RCM4000 RabbitCore
    - RCM3750 RabbitCore
    - RCM3600 RabbitCore
    - RCM3365 RabbitCore
    - RCM3200 RabbitCore
    - RCM3000 RabbitCore
    - **RCM3700 RabbitCore (Included in comparison.)**
    - RCM3400 RabbitCore
    - RCM3305 RabbitCore
    - RCM3100 RabbitCore
    - RCM2300 RabbitCore
    - RCM2100 RabbitCore
    - RCM2200 RabbitCore
    - RCM2000 RabbitCore
- SoftBaugh
    - USB430
    - **TCP430 (Included in comparison.)**
- Systronix
    - JStick Module
- Technologic Systems
    - TS7300
    - TS7200
    - TS7250
    - **TS7260 (Included in comparison.)**
    - TS5700
    - TS5600
    - TS5500
    - TS5400
    - TS5300
    - TS3400
    - TS3300
    - TS3200
    - TS3100
- WIZnet
    - EG-SR-7150MJ
    - IIM7300

Appendix B. Microcontroller Pricing Sources

The following screen captures show pricing information for the Microchip PIC18F6621 and Texas Instruments MSP430F149 obtained on 9 May 2006. For each product the lowest cost was used for comparison.



http://www.microchipdirect.com/productsearch.aspx?Keywords=PIC18F6621

Found **8** matches total. Items 1 thru 8 displayed.

Part Number: PIC18F6621-E/PT — PIC18F6621 view datasheet

Lead Count :	64	Pricing: (USD)	1-25	9.18	Availability :	Ship By: 13-Jun-2006
Package Type :	TQFP		26-99	8.94	Quantity:	
Temp Range :	-40C to +125C		100-999	8.69		
Packing Media :	Tray		1000-4999	8.19 *		
			5000-9999	7.90 *		
			10000+	6.89 *		

* microchip**DIRECT** account and approved quote required. Open a microchip**DIRECT** account today!

Part Number: PIC18F6621-I/PT — PIC18F6621 view datasheet

Lead Count :	64	Pricing: (USD)	1-25	8.31	Availability :	Ship By: 13-Jun-2006
Package Type :	TQFP		26-99	8.09	Quantity:	
Temp Range :	-40C to +85C		100-999	7.87		
Packing Media :	Tray		1000-4999	7.41 *		
			5000-9999	7.16 *		
			10000+	6.25 *		

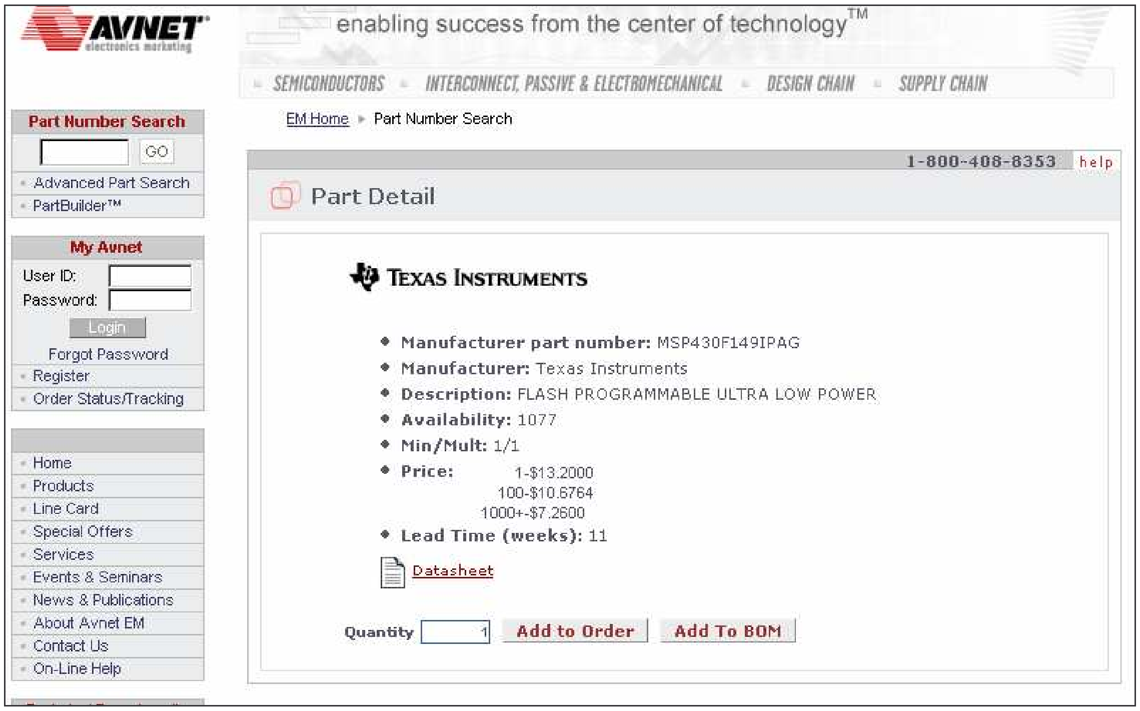
* microchip**DIRECT** account and approved quote required. Open a microchip**DIRECT** account today!

Part Number: PIC18F6621-I/PTG Pb — PIC18F6621 view datasheet

Lead Count :	64	Pricing: (USD)	1-25	8.31	Availability :	Ship By: 13-Jun-2006
Package Type :	TQFP		26-99	8.09	Quantity:	
Temp Range :	-40C to +85C		100-999	7.87		
Packing Media :	Tray		1000-4999	7.41 *		
			5000-9999	7.16 *		
			10000+	6.25 *		

* microchip**DIRECT** account and approved quote required. Open a microchip**DIRECT** account today!

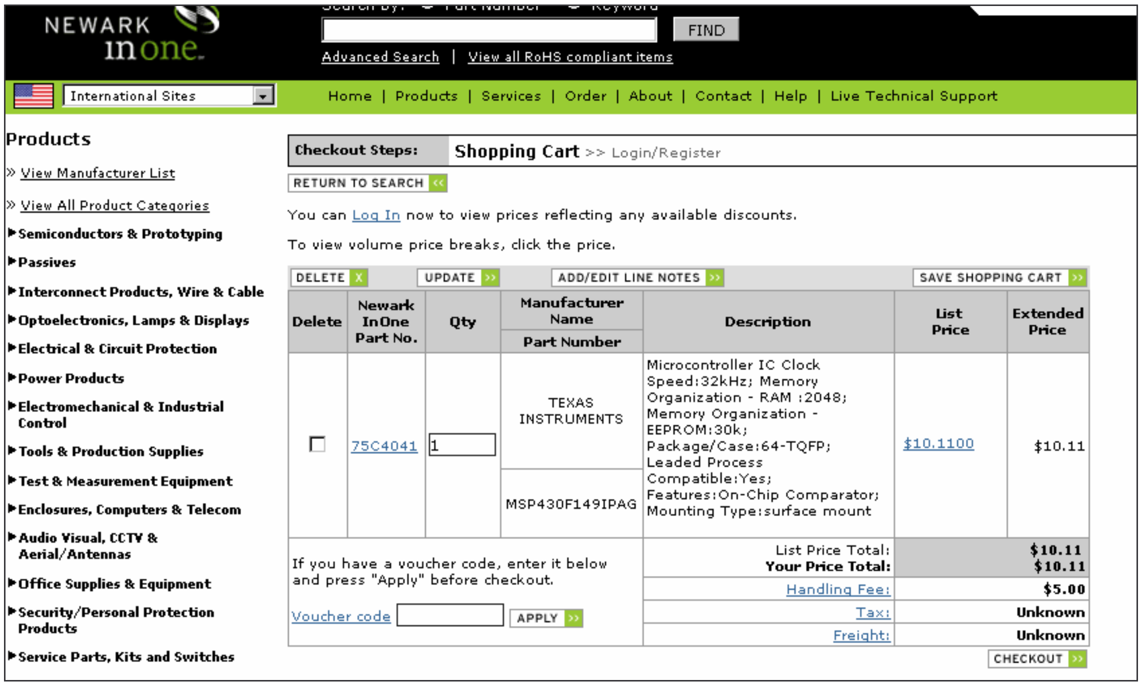
Figure 39: PIC18F6621 Pricing



- **Manufacturer part number:** MSP430F149IPAG
- **Manufacturer:** Texas Instruments
- **Description:** FLASH PROGRAMMABLE ULTRA LOW POWER
- **Availability:** 1077
- **Min/Mult:** 1/1
- **Price:** 1-$13.2000
  100-$10.6764
  1000+-$7.2600
- **Lead Time (weeks):** 11

Quantity 1

Figure 40: MSP430F149 Avnet Pricing


NEWARK inone
FIND
Advanced Search | View all RoHS compliant items
International Sites
Home | Products | Services | Order | About | Contact | Help | Live Technical Support
Products
» View Manufacturer List
» View All Product Categories
Semiconductors & Prototyping
Passives
Interconnect Products, Wire & Cable
Optoelectronics, Lamps & Displays
Electrical & Circuit Protection
Power Products
Electromechanical & Industrial Control
Tools & Production Supplies
Test & Measurement Equipment
Enclosures, Computers & Telecom
Audio Visual, CCTV & Aerial/Antennas
Office Supplies & Equipment
Security/Personal Protection Products
Service Parts, Kits and Switches
RETURN TO SEARCH
DELETE
UPDATE
ADD/EDIT LINE NOTES
SAVE SHOPPING CART
APPLY
CHECKOUT

Checkout Steps: **Shopping Cart** >> Login/Register

You can Log In now to view prices reflecting any available discounts.

To view volume price breaks, click the price.

Delete	Newark InOne Part No.	Qty	Manufacturer Name / Part Number	Description	List Price	Extended Price
☐	75C4041	1	TEXAS INSTRUMENTS / MSP430F149IPAG	Microcontroller IC Clock Speed:32kHz; Memory Organization - RAM :2048; Memory Organization - EEPROM:30k; Package/Case:64-TQFP; Leaded Process Compatible:Yes; Features:On-Chip Comparator; Mounting Type:surface mount	$10.1100	$10.11

If you have a voucher code, enter it below and press "Apply" before checkout.

Voucher code

List Price Total:	$10.11
Your Price Total:	$10.11
Handling Fee:	$5.00
Tax:	Unknown
Freight:	Unknown

Figure 41: MSP430F149 Newark Pricing

Appendix C. Schematics

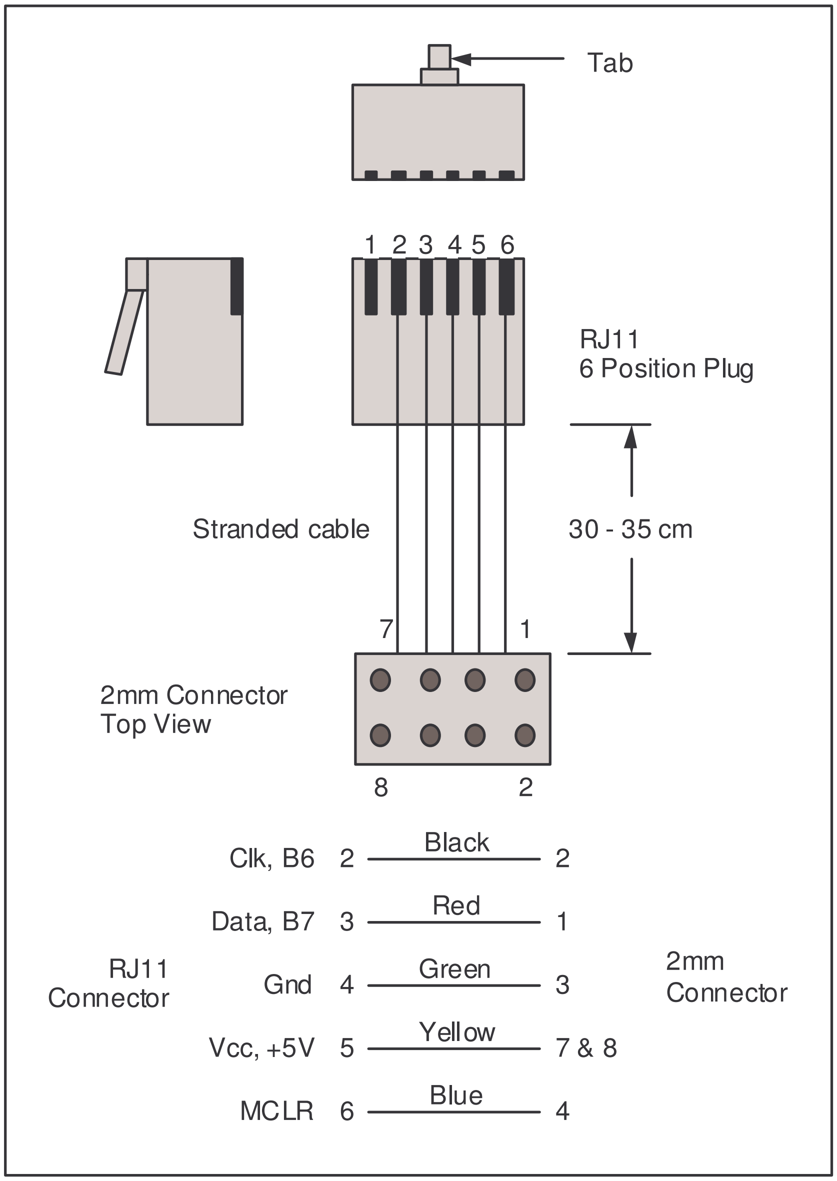


Figure 42: MACH-X to SBC65EC ICSP Adaptor Cable Schematic

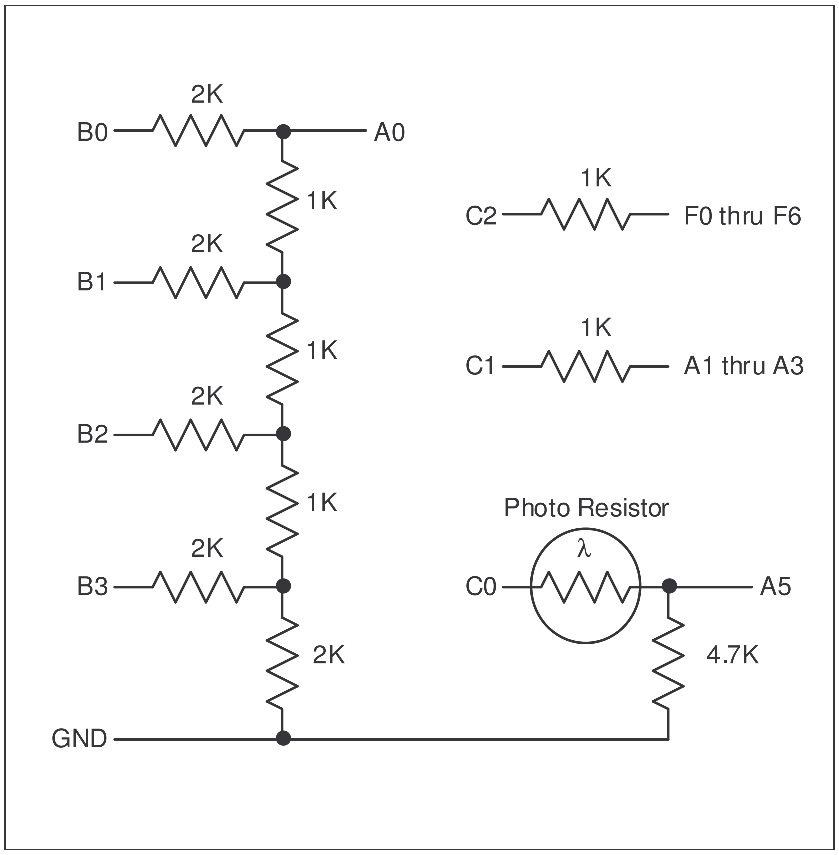


Figure 43: SBC65EC Expansion Board Schematic

Appendix D. Clockspeed Code Modifications

Code modifications of five different sections were required to run at a 10MHz clock speed. Those portions of the original code that needed modification (edited for readability) and the modified code are listed here for reference.

<u>Original Code:</u>

projdefs.h
```
    #define CLOCK_FREQ (40000000) //Hz
    //#define CLOCK_FREQ (20000000) // Hz
    //#define CLOCK_FREQ (10000000) // Hz
```

mxwebsrvr.c
```
    #if defined(__18CXX)    //Microchip C18 compiler
      #pragma config OSC = HSPLL, OSCS = OFF
      #pragma config PWRT = ON
      #pragma config BOR = OFF, BORV = 42
      #pragma config WDT = OFF
      //#pragma config CCP2MX = OFF
      #pragma config LVP = OFF
      #pragma config DEBUG = OFF
    #endif
```

delay.h
```
    #elif defined(MCHP_C18)
      #define  Delay10us(us)  Delay10TCYx(((CLOCK_FREQ/4000000)*us))
      #if (CLOCK_FREQ == 40000000)
        #define DelayMs(ms) Delay10KTCYx((((CLOCK_FREQ/1000)*ms)/40000))
      #elif (CLOCK_FREQ == 20000000)
        #define DelayMs(ms)  Delay1KTCYx((((CLOCK_FREQ/1000)*ms)/4000))
      #endif
```

delay.c
```
    void Delay10us(unsigned char us)
      {
      unsigned char i;
      while (us--)
        {
        #if (CLOCK_FREQ == 40000000) i = 13;
        #elif (CLOCK_FREQ == 20000000) i = 6;
        #else
        #error "Delay does not support current Clock Frequency"
        #endif
        //Delay 10us.  This takes 6 Instruction cycles per loop
        while (i != 0) {i--;}
        }
      }
```

appcgf.c
```
    #if (CLOCK_FREQ == 40000000)
        //Get USART BAUD rate setting
        switch (AppConfig.USART1Cfg.baud)
    (this section of code deleted)
    #else
    #error "appcfgUSART() does not support selected clock frequency"
    #endif
```

Modified Code:

projdefs.h

```
    //#define CLOCK_FREQ (40000000) //Hz
    //#define CLOCK_FREQ (20000000) // Hz
    #define CLOCK_FREQ (10000000) // Hz
```

mxwebsrvr.c

```
    #if defined(__18CXX)    //Microchip C18 compiler
      //#pragma config OSC = HSPLL, OSCS = OFF
      #pragma config OSC = HS, OSCS = OFF // added to drop clock to 10MHz
      #pragma config PWRT = ON
      #pragma config BOR = OFF, BORV = 42
      #pragma config WDT = OFF
      //#pragma config CCP2MX = OFF
      #pragma config LVP = OFF
      #pragma config DEBUG = OFF
    #endif
```

delay.h

```
    #elif defined(MCHP_C18)
      #if (CLOCK_FREQ == 40000000)
         #define Delay10us(us)  Delay10TCYx(((CLOCK_FREQ/4000000)*us))
         #define DelayMs(ms)  Delay10KTCYx((((CLOCK_FREQ/1000)*ms)/40000))
      #elif (CLOCK_FREQ == 20000000)
         #define Delay10us(us)  Delay10TCYx(((CLOCK_FREQ/2000000)*us))
         #define DelayMs(ms)  Delay1KTCYx((((CLOCK_FREQ/1000)*ms)/20000))
      #elif (CLOCK_FREQ == 10000000)
        #define Delay10us(us)  Delay10TCYx(((CLOCK_FREQ/1000000)*us))
        #define DelayMs(ms)  Delay1KTCYx((((CLOCK_FREQ/1000)*ms)/10000))
        #endif
```

delay.c

```
    void Delay10us(unsigned char us)
      {
      unsigned char i;
      while (us--)
        {
        #if (CLOCK_FREQ == 40000000) i = 13;
        #elif (CLOCK_FREQ == 20000000) i = 6;
        #elif (CLOCK_FREQ == 10000000) i = 3;
        #else
        #error "Delay does not support current Clock Frequency"
        #endif
        //Delay 10us.  This takes 6 Instruction cycles per loop
        while (i != 0) {i--;}
        }
      }
```

appcgf.c

```
    #define BAUD_RATE1 (57600) // bps
    #define SPBRG1_VAL ((((CLOCK_FREQ/BAUD_RATE1)/4) - 1) & 0xff)
    #define SPBRGH1_VAL (((((CLOCK_FREQ/BAUD_RATE1)/4) - 1) >> 8) & 0xff)
```