2390 EMAC, Way
Carbondale, IL 62902
Phone: (618) 529-4525
Fax: (618) 457-0110
http://www.emacinc.com

SOM-400EM Users Manual

SOM-E400 Users Manual rev1.1

Table of Contents

1. Introduction

This document describes EMACs SOM-400EM System on a Module (SoM). The SOM-400EM is Based on the 8051 code compatible Dallas DS80C400 TINI processor. The Tiny InterNet Interface (TINI) processor is a Java programmable processor that is ideal for use in Internet appliances and Web based applications.

This 8-bit 8051 code compatible processor can run at 75Mhz and has an Ethernet MAC built-in along with 3 serial ports. It can directly access 16 Meg of memory and has a Unix type OS that features a complete file system.

The only drawback to this processor was that it does not have a hardware SPI port. To overcome this drawback, EMAC designed a PLD SPI engine that connects directly to the processors data bus. This allows the EMAC SOM-400EM to meet customer SPI speed requirements, that a bit-banged SPI could not. EMAC also added a programmable oscillator and 10 general-purpose digital I/O lines to this SoM.

Using the same SODIMM form-factor that Dallas used on their reference board, EMAC then added additional higher speed memory, an Ethernet PHY, a hardware SPI port, and digital I/O lines. EMAC also wrote a custom native SPI driver to support the hardware SPI port. A special version of the board can be purchased that is backwards compatible with the Dallas reference board.

The SOM-400EM is designed to plug into a carrier board that contains all the connectors and any custom I/O required for the application. With this System On Module approach a semi-custom hardware platform can be developed in as little as a month. A development carrier board (SOM-100ES) is available from EMAC, allowing the user to immediately start coding their application.

The features of the SOM-400EM are as follows:

1.1 Features

- Small, 144 pin SODIMM form factor (2.66" x 1.5")
- IPv4/6 Fast Ethernet with on-board PHY
- 3 serial port with handshake
- 1 non-isolated CAN port
- Up to 2 Meg. of low power battery backed RAM
- 2 meg of Flash
- Nonvolatile RAM/File System
- Battery backed Real Time Clock
- 1-Wire Network provision
- High-speed math accelerator for 16/32-bit multiply and divide
- Typical power requirement of about 1.5 watts
- Robust FREE Java development tools

1.2 Options

ON-BOARD OPTIONS

- **Memory:** 2 Meg of RAM.
- **Software:** ANSI C compiler.

1.3 Other Options

- **Carrier Board:** A board designed to accommodate the SOM-400EM and other SODIMM SOMs providing additional I/O and connector terminations.

2. Hardware

2.1 Specifications

- **VOLTAGE REQUIREMENTS:** +3.3 Volts DC +/- 5%
- **CURRENT REQUIREMENTS:** Typical 3.3 Volts @ 200 ma.
- **OPERATING TEMPERATURE:** 0 - 70 degrees Centigrade, humidity range without condensation 0% to 90% RH.
- **DIGITAL I/O: 5** dedicated Digital Inputs and 5 dedicated Digital Outputs with 25 ma. drive capability.
- **TIMER/COUNTERS:** 4, 16-bit timers/counters with one up/down timer, capture, and baud-rate generation features

2.2 SoM Dependent Jumpers

The following jumpers are located on the SOM-100ES Carrier board and are SoM dependent. When used with a SOM-400EM they are defined as follows:

2.2.1 JP7

DTR Reset This jumper is used to allow JavaKit to reset the SOM-400EM through the serial port. This jumper should be enabled when using JavaKit. (default position 2 – 3, disabled).

2.2.2 JP9

Netboot This jumper allows the SOM-400EM to connect to a TFTP server and to automatically download a program and execute it (default position 2 – 3, disabled –> P5.3 to Vcc).

2.2.3 JP11

Quietboot This jumper causes the SOM-400EM to bypass the bootloader and proceed to load the operating system (default position 1 – 2, enabled –> P1.7 to Gnd).

2.3 On-Board PLD Programming

A JTAG port provides a programming interface to the PLD. This port is accessible through the SODIMM connector. The PLD comes preprogrammed with the SPI engine from EMAC. This PLD can be custom programmed to provide other functionality if required.

The PLD is connected to the DS80C400 processor using the data bus and I/O chip select (P4.3/*MCE3).

Table 1: JTAG CONNECTIONS

SODIMM Pin	Signal
137	JTAG-TDI
138	JTAG-TDO
139	JTAG-TCK
140	JTAG-TMS
141	3.3 VCC
143	GND

2.4 On-Board PLD Based General Purpose Digital I/O

These input and output lines provide connections for relays, switches, etc. They are connected to the PLD of the SOM-400EM. PLD lines X0 - X4 a dedicated input port whose input lines should not exceed 5 Vdc. The dedicated output port lines PY0 – PY4 can drive 25 ma. loads. These Digital I/O lines are accessible through the SODIMM connector as defined in the table below. Also available is a software programmable status LED. This LED is connected to processor port P5.2/T3 and can be lit by setting the port line low.

Table 2: PLD BASED DIGITAL I/O

SODIMM Pin	Signal
125	PLD-X0
126	PLD-X1
127	PLD-X2
128	PLD-X3
129	PLD-X4
130	PLD-Y0
131	PLD-Y1
132	PLD-Y2
133	PLD-Y3
134	PLD-Y4

The PLD is connected to the DS80C400 processor using its data bus and control lines. The PLD decodes two MPU address bus lines A0 and A1, into four addresses. The decoding is qualified by a device select input, active low. Therefore, the actual addresses of each internal register is the Base address (A.K.A., ~chip enable), plus A0 and A1. To access PLD Port X, perform a Read at address 60002. To access PLD Port Y, perform a write with at address 60002. They are further qualified by bus signals ~READ (A.K.A. ~PSEN), and ~WR. All addressing shown is listed in HEX. They are:

Table 3: PLD BASED DIGITAL I/O

Hex Address	Function	Port
60002	Read	X
60002	Write	Y

2.5 SPI

The SOM-400EM provides a configurable SPI engine. The SPI can be configured through software to change clock frequency and polarity. Also provided within the SPI PLD are two programmable SPI chip selects. These two lines can be externally decoded to allow up to four SPI slave devices. A native Java driver is provided for the SPI port.

Table 4: SPI

SODIMM PIN	Signal
120	SPI-CLK
121	SPI-MOSI
122	SPI-MISO
123	SPI-CS0
124	SPI-CS1

2.5.1 SPI Operation

This customized SPI operates as a MASTER only. It is similar in basic functionality to the Motorola SPI, but there are some important differences. They are:

- An SPI transaction is always 8 bits.
- The SPI data direction is fixed at MSB out first LSB last.
- The SPI has four internal states per clock output cycle.
- A transmit and receive occur simultaneously, as does the Motorola SPI.
- An SPI transaction is automatically started upon a data write to the SPI data register.
- Completion of an SPI transaction is signaled on Processor pin 75 (P3.2/INT0).
- The MOSI output is tri-stated between SPI active cycles.

The transaction completed status hardware line is set high after the SPI transaction is finished, and cleared when data is read out to the data bus. If another write should occur without a prior read, this pin will snap back low while the SPI again functions, and return high again upon the new completion. The readable data that was not read will be lost and replaced.

This SPI is NOT buffered. The same shift register that transmits and receives data is the data I/O register. One CANNOT read the outgoing data prior to a transmit cycle, as the very act of the initial write will cause the SPI to run through it's cycle, and the data left behind in this register is whatever was simultaneously shifted in at that time.

The encoded external device selected is NOT AUTOMATICALLY synchronized to the SPI cycle. The encoded address must be written via a carefully masked control address write transaction, prior to starting an SPI cycle (by a data write), and, if need be, cleared by a separate control address write transaction.

The SPI has two speed ranges (pre-scales). The default is divided by one (fast). If a bit is set in the control register, then a divide by four (slow) prescale setting is in place. As the SOM-400EM clock chip currently supplies a base clock of 8.00 MHz to the PLD, and there are four internal states per cycle, the fast SPI clock output is therefore 2.00 MHz. The slow SPI clock setting is 500 KHz. The clock chip can custom factory programmed to allow for just about any frequency.

As the SPI settings of clock polarity, phase, device select, and prescale are all part of one six bit data register which is not readable, the software engineer must keep a copy of everything written to this control register, and carefully mask bits in and out of it whenever writes are made to this control port, so as not to disturb various settings while changing device select addresses, etc.

The PLD is connected to the DS80C400 processor using its data bus and control lines. The PLD decodes two MPU address bus lines A0 and A1, into four addresses. The decoding is qualified by a device select input, active low. They are further qualified by bus signals ~READ (A.K.A. ~PSEN), and ~WR. All addressing shown is listed in HEX. They are:

Table 5: PLD SPI REGISTERS

Address	Function	Description
60000	Read	Data
60000	Write	Data
60001	Read	No Function
60001	Write	Configuration

Table 6: SPI CONFIGURATION REGISTER DESCRIPTION

Bit	Description	Default
0	Encoded external SPI device select CS0	0 (not selected)
1	Encoded external SPI device select CS1	0 (not selected)
2	SPI clock polarity bit. Same as Motorola SPI bit CKPOL	0 (positive)
3	SPI clock phase bit. Same as Motorola SPI bit CKPHA	0 (no shift)
4	SPI clock frequency prescaler (divide by 1 or by 4)	1 (divide by 1)
5-7	No Function	

2.6 Chip Selecting External Devices

There are several memory chip selects available on the SOM-400EM that can be used to select external devices. Depending on the SOM-400 options some of these chip selects maybe utilized on the SOM-400EM itself. If utilizing the SOM-100ES carrier board, it may also uses several of these chip selects in order to provide additional I/O functionality. The table below outlines the chip selects and there functions.

Table 7: CHIP SELECTS

SODIMM Pin	Signal	SOM-400EM	SOM-100ES	Memory Address
NC	*CE0	RAM Select	Not Used	00000
100/101	*CE1	Not Used	Not Used	20000
NC	*CE2	Flash Select	Not Used	40000
98	*CE3	SPI Engine	Not Used	60000
108	*CE4	Not Used	PLD CS	80000
107	*CE5	Not Used	COM3 DSR	A0000
106	*CE6	Not Used	COM3 DTR	C0000
105	*CE7	Not Used	D/A *LDAC	E0000

2.7 COM 0 UART

The SOM-400EM provides one dedicated UART Serial 0 which has software configurable baud rates. By default no handshake lines are available for COM 0. This COM port, in conjunction with the processor Reset line, is normally used by the Bootloader and SLUSH. It can however by utilized by application software as well.

Table 8: COM 0

SODIMM Pin	Signal	Description
71	P3.0/RXD0	RxD
73	P3.1/TXD0	TxD

2.8 COM 1 UART

The SOM-400EM provides one COM 1 UART, which has software configurable baud rates. Handshake Lines are implemented by the use of port lines P5.6 and P5.7.

Table 9: COM 1

SODIMM Pin	Signal	Description
36	P1.3/TXD1	TxD
38	P1.2/RXD1	RxD
78	P5.7/*PCE3	CTS-IN
82	P5.6/*PCE2	RTS-OUT

2.9 COM 2 UART

The SOM-400EM provides one COM 2 UART, which has software configurable baud rates. Handshake Lines are implemented by the use of port lines P1.4 – P 1.6, P3.3, P5.5 and P6.1.

Table 10: COM 2

SODIMM Pin	Signal	Description
102	P6.7/TXD2	TxD
103	P6.6/RXD2	RxD
30	P1.4/INT2	DCD-IN
32	P1.5/*INT3	DTR-OUT
39	P1.6/INT4	RTS-OUT
76	P5.5/*PCE3	RI-IN
79	P3.3/*INT1	CTS-IN
107	P6.1/*CE5	DSR-OUT

2.10 CAN I/O

The SOM-400EM provides a single CAN 2.0B port. In order to use the CAN on a CAN Bus Network an external CAN interface chip must be used.

Table 11: CAN

SODIMM Pin	Signal
95	P5.1/CAN-RX
96	P5.0/CAN-TX

2.11 1-Wire Network

The Dallas DS80C400 features an on chip 1-Wire Master. This network can be used to interface to a number of peripheral device chips available from Dallas. This network utilizes a driver circuit with an active pull-up. The pull-up is turned on by applying a low to the OWSTP pin. The pull-up voltage is controlled externally on pin number 118 of the SODIMM. Refer to the DS80C400 user manual for additional information.

Table 12: 1-Wire

DS80C400 Pin	Signal	Description
99	OW	1-Wire
100	OWSTP	Active Pull-up

2.12 Ethernet

The SOM-400EM provides a 10/100 Ethernet port. The MAC located within the DS80C400 processor and the PHY are both included on the module. All that is additionally required for a complete 10/100 BaseT connection is the magnetics, an RJ45 connector, a few passive components and status LEDs if required. Refer to the EMAC Carrier board reference design for further details.

The Dallas reference design does not include the Ethernet PHY on the module and assumes that this will be externally connected. A special version of the SOM-400EM can be ordered without the PHY for compatibility with the Dallas module.

A complete TCP/IP stack is provided within firmware of the DS80C400 chip. This stack is automatically utilized within the TINI Java environment. To utilize the stack from within a C program refer to Dallas's application notes regarding this subject.

Table 13: Ethernet

SODIMM Pin	Signal	Description
89	ETH-LED1	Link LED
90	ETH-LED2	Receive Status LED
91	TPOP	Ethernet TX+
92	TPON	Ethernet TX-
93	TPIP	Ethernet RX+
94	TPIN	Ethernet RX-

2.13 Real Time Clock

The SOM-400EM utilizes either the Dallas DS2502 or DS2417 Real Time Clock (RTC). Both of these clock chips use a 32-bit Clock Counter. An external battery should be connected to keep from losing the time when power is removed from the SOM-400EM.

Table 14: DS1672 RTC

DS80C400 Pin	Signal	DS1672 Pin	Signal
24	P3.4/T0	5	SDA
25	P3.5/T1/CLK0	6	SCL

Table 15: DS2502 RTC

DS80C400 Pin	Signal	DS2502 Pin	Signal
99	OW (1-Wire)	2	1-Wire

3. Software

3.1 Introduction

The SOM-400EM can be programmed in Java, C, and theoretically Assembler. While there is support for programming the unit in C (using the Keil compiler available from EMAC) the vast majority of the users will make use of Java.

There are a number of advantages to programming the SOM-400EM in Java. Some these advantages are:

- Programming Ease
- Extensive Network Support
- Robust Free Development Tools
- Numerous Open Source Applications (e.g. Slush, Web Server, Serial Server, etc.)

The only drawback to programming in Java versus C is the execution speed. Java is an interpreted language and as such runs considerably slower than C. Whether speed is an issue depends on your application. Most network applications run fine on the SOM-400EM. If additional speed is required the user can copy his code to RAM and run with less wait states/higher clock speed.

3.2 Setting Up To Run Java

3.2.1 Requirements

The minimum hardware and software requirements are:

- TINI Hardware Requirements
  - SOM-400EM TINI Module
  - SOM-100ES Carrier Board or equivalent

- TINI Software Requirements
  - TINI Software Development Kit - Version 1.1 or later
  - JDK (Java Development Kit) - from Sun Microsystems or Jbuilder 8 – from Borland or equivalent (1)
  - Java Communications API - from Sun Microsystems or http://www.rxtx.org

- Development System Requirements
  - Operating system meeting requirements of the JDK from Sun Microsystems
  - 1 RS232 (COM) port Œ115200 baud recommended
  - RS232C serial cable DB9 male to DB9 female - Radio Shack Catalog Number 26-117 or equivalent
  - Crossover Ethernet cable - Radio Shack Catalog Number 950-0368 or equivalent for direct connection to host computer, a straight through cable should be used for connecting to a router or hub
  - 5 – 12 VDC Power Supply capable of supplying 250mA (EMAC part # T600-50-0 or equivalent)

3.2.2 Quick Start

1. Prior to beginning, verify power is not connected to the SOM-100ES Carrier board.
2. Insert the SOM-400EM Module into the 144-pin connector on the SOM-100ES Carrier board.
3. Attach an RS232 serial cable to the connector labeled CN1.
4. Connect the serial cable to a serial port on your PC.
5. Next, attach the crossover Ethernet cable between the PC and the SOM-100ES Carrier board.
6. Connect the power adapter to your SOM-100ES Carrier board.
7. Plug the power adapter into a wall socket.

NOTE: If the TINI sockets board is not being connected directly to a network interface card, but instead to a separate network port on a hub or switch, a straight through Ethernet patch cable should be used.

3.2.3 Java Development Kit Software Setup

First, download and install the Java Development Kit and Java Communications API from Sun Microsystems. Installation instructions and example applications are included in both packages. You should verify correct installation by running the `BlackBox` example provided with the Java Communications API.

3.2.3.1 Loading the TINI Runtime Environment

This step has been performed by the factory and should not need to be done unless the SOM-400EM's Flash has been corrupted. If this is the case then download the TINI Software Development Kit from http://www.maxim-ic.com/TINI. The download file should be unzipped to a location of your choice on your hard drive (`<TINI SDK Install Dir>`). To begin using the SOM-400EM Module, the TINI runtime environment must first be loaded into the flash of the module. Loading the runtime environment requires executing a program called JavaKit. Application binaries are loaded through the serial port of the TINI module. If using JavaKit be sure jumper JP7 is enabled (position 1 – 2).

JavaKit is a special terminal program that is used to access the bootloader as well as just communicate with the SOM-400EM. If accessing the bootloader is not required then any terminal program such as HyperTerminal will suffice.

3.2.3.2 Running JavaKit

Note: The SOM-400EM comes with the FLASH configured with the Runtime Environment and Slush loaded. Steps 6 – 10 below are only required if the Flash has been corrupted and needs to reloaded.

1. To run JavaKit, open a command shell or DOS prompt on your PC. Change to the `<TINI SDK Install Dir>\bin directory`.
2. Next, `type java -classpath tini.jar JavaKit -400 -flash 40` at the command prompt and press return to run the JavaKit application. If you have the Java Development Kit and the Java Communications API properly installed, JavaKit should appear on the screen.
3. Select the port name where the TINI SOM400-EM is attached. Press the Open Port button. The default baud rate of 115200 should be used on all platforms where it is supported.
4. After opening the port, press the Reset button. The loader prompt should be displayed on the JavaKit screen.
5. If the message is not displayed, verify the cable connections and check that the correct port name is selected. If everything appears to be attached correctly, read the Running_JavaKit.txt file contained in the TINI SDK for more detailed information.
6. At the JavaKit prompt, type **`B0`** and press return followed by **`F0`** and return. This clears the RAM of the TINI Verification Module. Any time a new runtime environment is loaded the memory should be cleared.
7. Now click on the File menu in JavaKit and select Load File. A file dialog will appear on the screen. Browse to the `<TINI SDK Install Dir>\bin directory` and select the `tini_400.tbin` file and click the Open button. Load Complete will be displayed on the screen when the firmware is loaded, this may take up to 2 minutes.
8. To load the command shell, click on the File menu and select Load File again. Select the `slush_400.tbin` file and click on the Open button. When Load Complete appears, slush has been successfully loaded into the flash of the TINIm400 Verification Module
9. Press the reset button and the JavaKit loader prompt will be displayed.
10. At the JavaKit prompt, type **`E`** and hit return. Slush will begin booting and the Slush boot output will print on the JavaKit screen.
11. Press <`return`> and a Slush login prompt will be displayed. Type in the username `root` and press <`return`>. The initial password is `tini`.
12. You should now be logged into Slush. To view a list of valid shell commands, type help at the command prompt. Detailed information on specific commands, such as `dir`, can be obtained by typing `help` followed by the command.

3.2.3.3 Quick Start Using HyperTerminal

1. Select the port name where the TINI SOM400-EM is attached, using Direct Connect. The baud rate of 115200 should be used with No Parity, 1 Stop Bit, and No Handshaking.

2. Press the `<return>` Key and a Slush login prompt will be displayed. Type in the username `root` and press `<return>`. The initial password is `tini`.

3. You should now be logged into Slush. To view a list of valid shell commands, type help at the command prompt. Detailed information on specific commands, such as `dir`, can be obtained by typing `help` followed by the command.

3.2.3.4 Network Setup

One of the primary features of the TINI Platform is its ability to access the network. The slush command to configure network access is `ipconfig`. Typing `ipconfig -h` at the command prompt will display detailed configuration options. The -h option may be used with any slush command to view options. The simplest method of connecting to the network is with DHCP, assuming it is supported on your network. To enable DHCP, type `ipconfig -d` and press enter. Once the TINI Verification Module has successfully leased a network address, a message is displayed on the slush prompt indicating success. Note: the DS80C400 ROM supports IPv6 for networking. By default, it is always on and configured automatically.

If your network does not support DHCP, then it is necessary to use a static IP address. First, you must get a valid network address and subnet mask from the network administrator. To set the IP address to a static value, use `ipconfig -a x.x.x.x -m y.y.y.y` where the IP address is denoted by x.x.x.x and the subnet is y.y.y.y. For example, to set your IP address to 192.168.0.1 with a subnet of 255.255.255.0, type `ipconfig -a 192.168.0.1 -m 255.255.255.0` at the slush command prompt and press enter.

A good way to test your network settings is to use the ping command. Either attempt to ping the TINI Verification Module from the host computer or ping the host computer from the TINI Verification Module using the slush ping command.

3.3 Running Your First Application

The TINI SDK contains many example applications. Each example application has batch files for compiling and converting it to a binary file for execution on the TINI Runtime Environment. A very simple application included with the TINI SDK is Blinky. This application blinks an LED (DS1) on the SOM-400EM. To transfer the application to your TINIm400, open a command prompt on your host computer and change to the `<TINI SDK Install Dir>\examples\Blinky\bin directory`. This directory contains a file called `Blinky.tini`. From the command prompt on the host computer type `ftp`. The commands below will connect to the TINI and transfer the `Blinky.tini` file to the SOM-400EM's file system. Replace the IP address used below with the IP address you set earlier in Slush. The default password for ftp and telnet on the TINI Runtime Environment is `tini`.

```
ftp> open 192.168.0.1
Connected to 192.168.0.1.
220 Welcome to slush. (Version 1.1) Ready for user login.
User  192.168.0.1:(none)): root
331 Password Required for root
Password:
230 User root logged in.
ftp> bin
200 Type set to Binary
ftp> put Blinky.tini
200 Command successful.
150 BINARY connection open, putting Blinky.tini
226 Closing data connection.
ftp: 514 bytes sent in 0.00Seconds 514000.00Kbytes/sec. ftp> bye
221 Goodbye The file now exists in the TINI file system.
```

Next, connect to your TINIm400 using telnet on your PC. Verify the existence of the Blinky.tini file using the Slush command `ls`. To run the `Blinky.tini` file, type `java Blinky.tini` followed by enter. The Blinky program executes and the LED should begin a steady blink.

3.4 SPI400EM SPI Driver Documentation

This document explains the use of the SPI400em driver for the EMAC SOM-400EM DS400 module.

3.4.1 Structure

The SPI400EM driver consists of 2 parts, the native spi400em.tlib library, and the SPI400EM.java file. The core code is implemented with native methods for speed, and is located within a tlib file, which is short for tini library.

The SPI native methods are used in JAVA through the SPI400EM object provided by the JAVA file. The a51 assembly instructions used to interface with the SPI IC are contained in the tlib library.

To use this code in a project, the JAVA file should be included in the project. By creating a SPI400EM object in the main JAVA application a tini JAVA file can be created, which has the extension .tini. When this tini file is executed by the JAVA interpreter it will look for a file called spi400em.tlib in the same directory as the tini file. If this file is not present the program will generate an unsatisfied link error and exit.

3.4.2 Interface

The EMAC SPI400EM driver has been designed to closely mirror the interface of the old bit-banged SPI used on the Dallas TINI reference board. Some configuration options have changed because different hardware is being used, and its capabilities have changed since the original implementation

Configuration is done when the object is created, and dynamically by methods within the object. Available options include, clock divider, SS line number, CPOL and CPHA.

Transmit and receive always happens simultaneously by means of the xmit method. The xmit method transmits an array of bytes out of the SPI port and simultaneously fills the same array with incoming data.

The name of the class that provides the Java interface is called SPI400EM.java. Please refer to the SPI400EM.html file for further details.

4. More Information

Details of the TINI Platform are on the http://www.maxim-ic.com website. The ”TINI Specification and Developers Guide” is an invaluable resource when developing with the TINI platform and can be downloaded from the Maxim web site. Chapter 2 is dedicated to getting started with the TINI Platform and includes a detailed description of building and running several small example applications. Included with your SOM-400EM is a CD that contains most of what is required to get started.

4.1 Additional Information and Links

Java Development Kit: http://java.sun.com/j2se

JavaBuilder: http://www.borland.com/

Java Communications API (Java COMM)

- Windows/Solaris: http://java.sun.com/products
- Linux: http://www.rxtx.org

Dallas Semiconductor: http://maxim-ic.com/tini

Sales and Customer Service: http://www.emacinc.com

Product Information: http://www.emacinc.com.com/som/som-400em.htm

TINI and 1-Wire are registered trademarks of Dallas Semiconductor.
SPI is a trademark of Motorola, Inc.
Java is a trademark of Sun Microsystems.
Internet Explorer is a trademark of Microsoft Corp.

5. Appendix A

5.1 SOM-400EM SODIMM Pinout

	Note: '*' indicates pin change from Dallas TINI module						
PIN	**DESCRIPTION**	**PIN**	**DESCRIPTION**	**PIN**	**DESCRIPTION**	**PIN**	**DESCRIPTION**
1	Ground	37	A12	73	P31	109	A5
2	Ground	38	P12	74	P53	110	NC
3	Vcc 3.3	39	P16	75	P32	111	A6
4	Vcc 3.3	40	P11	76	nPCE1	112	NC
5	A13	41	P17	77	NC	113	A7
6	NResetIn	42	P10	78	nPCE3	114*	CLK 1 (8 Mhz)
7	A16	43	nRstOut	79	P33	115*	CLK 2 (200 Khz)
8*	NC	44	NEA	80	nPCE0	116	Local1W
9	A15	45	CRS	81	NC	117*	CLK 3 (14.3818 Mhz)
10	A8	46	TxEn	82	nPCE2	118	Vpullup
11	A14	47	COL	83	nWr	119*	Vbat In
12	A9	48	RxEr	84	NC	120*	SPI ~SCK 1
13	A17	49	RxDv	85	nRd	121*	SPI MO 1
14	A11	50	MDC	86	NC	122*	SPI MI 1
15	A20	51	MDIO	87	P52	123*	SPI CS0
16	nPSEn	52	Ground	88*	Ethernet LED 0	124*	SPI CS1
17	A19	53	Ground	89*	Ethernet LED 1	125*	PLD X0
18	A10	54	TxD3	90*	Ethernet LED 2	126*	PLD X1
19	D6	55	TxD2	91*	Ethernet Tx+	127*	PLD X2
20	D7	56	TxD1	92*	Ethernet Rx+	128*	PLD X3
21	D5	57	TxD0	93*	Ethernet Tx-	129*	PLD X4
22	D3	58	Ground	94*	Ethernet Rx-	130*	PLD Y0
23	D4	59	Ground	95	P51	131*	PLD Y1
24*	NC	60	TxClk	96	P50	132*	PLD Y2
25	D2	61	RxClk	97	A18	133*	PLD Y3
26	A0	62	Ground	98	nCE3	134*	PLD Y4
27	D1	63	Ground	99	NC	135*	NC
28	A4	64	RxD0	100*	nCE1	136*	NC
29	D0	65	RxD1	101*	nCE1	137*	JTAG TDI
30	P14	66	RxD2	102	P67	138*	JTAT TDO
31	A3	67	RxD3	103	P66	139*	JTAG TCK
32	P15	68	Ground	104	A21	140*	JTAG TMS
33	A2	69	Ground	105	nCE7	141	Vcc 3.3
34*	NC	70	NC	106	nCE6	142	Vcc 3.3
35	A1	71	P30	107	nCE5	143	Ground
36	P13	72	ALE	108	nCE4	144	Ground

5.2 SOM-400EM SODIMM Pinout Differences From Dallas TINI Module

Pins needed	#of Pins	Notes	Pin on the TINI Module
SPI PINS	5	spi bus pins, from the PLD	
~SCK_1		was NC	120
MO_1		was NC	121
MI_1		was NC	122
SPI_CS0		was NC	123
SPI_CS1		was NC	124
Ethernet Pins	7	Led pins provide cfg as well	
Eth_Led0		was NC	88
Eth_Led1		was NC	89
Eth_Led2		was NC	90
Eth_Tx+		was NC	91
Eth_Rx+		was NC	92
Eth_Tx-		was NC	93
Eth_Rx-		was NC	94
JTAG pins	4	JTAG daisy chain pins	
JTAG_TDI		was NC	137
JTAG_TDO		was NC	138
JTAG_TCK		was NC	139
JTAG_TMS		was NC	140
CLOCK PINS	3	Osc. Taps	
Clock1		was NC	114
Clock2		was NC	115
Clock3		was NC	117
NEW NC'S	6		
NC		was Dram2En	8
NC		was Dram1En	34
NC		was FlashEn	24
NC		was nCE2	99
NC		was nCE1	100
NC		was nCE0	101

Pins needed	#of Pins	Notes	Pin on the TINI Module
PLD_GPIO'S	10		
PLD_X0		was NC	125
PLD_X1		was NC	126
PLD_X2		was NC	127
PLD_X3		was NC	128
PLD_X4		was NC	129
PLD_Y0		was NC	130
PLD_Y1		was NC	131
PLD_Y2		was NC	132
PLD_Y3		was NC	133
PLD_Y4		was NC	134
BATTERY			
Vbat_In	1	was NC	119