

μ CAN2 Technical Reference

Document Revision 0.2



■ *A Complete
Reference to Using
& Programming the
Dallas High Speed Micro
Development
System μ CAN2*

μ CAN2

LIMITED WARRANTY

The information in this manual is subject to change without notice and does not represent a commitment on the part of Systronix, Inc. Systronix, Inc. makes no warranty, express or implied, for the use or misuse of its products, which are provided with the understanding that you, the user, will determine fitness for a particular application. Systronix assumes no responsibility for any errors which may appear in this manual. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose other than the purchaser's personal use without the written permission of Systronix, Inc.

Systronix reserves the right to revise this documentation and the software and hardware described herein or make any changes to the specifications of the product described herein at any time without obligation to notify any person of such revision or change.

TRADEMARKS

Systronix is a registered trademark of Systronix Inc, INT_EL and Intel are registered trademarks of Intel Corporation, Microsoft and MS-DOS are registered trademarks of Microsoft Corporation.

Systronix[®], Inc.
555 South 300 East #21
Salt Lake City, UT 84111
TEL: 801-534-1017
FAX: 801-534-1019
Internet: www.systronix.com
email: info@systronix.com

Copyright © 1999 by Systronix[®], Inc.
All rights reserved.

printed November 9, 1999

CREDITS and a BIT of RAMBLING FROM the AUTHOR

This manual was created using Corel WordPerfect 8.0 on an NT4 workstation. Schematics and the HSM/550 circuit board were created with Accel EDA and Scott Kendall. Prototype build and production managed by Jared Wyckoff. Network Services by Christopher Robin. Postscript output was obtained from an Apple LaserWriter IINT, PDF output using Acrobat 4.0.

- Bruce Boyes, Systronix, Inc.

μCAN2
*Technical
Reference*

Systronix, Inc.
Complete Solutions for Rapid Development
of Embedded Control Systems

Document Revisions

1. 0.1 start
2. 0.1a August 5, 1999 (10:05AM) updating loader and memory map specs
3. 0.2 August 30, 1999 updating based on changes to first prototype boards.

Table of Contents

PLEASE READ THIS FIRST	1
Errata & Anomalies	1
UCAN2 PCB Rev A	1
Terminology in This Manual	1
What You Need to Use uCAN2	2
QUICK START - STEP BY STEP	3
What you need	3
Step-by-Step Procedure	3
DETAILED DESCRIPTION	7
Philosophy and Purpose of uCAN2	7
Board Versions	7
Features	8
Loader and PLD Options and Updates	9
Systronix Web Site & Forum	9
Getting Technical Support	9
Installing and Using the Windows RAD51 IDE & Assembler	10
uCAN2 MEMORY	11
uCAN2 Memory Map in Load and Run Mode	11
Memory Backup	11
Nonvolatile System Parameters and Unique Network ID	11
Load Mode	11
RUN mode	12
Serial Loader 100% Non-intrusive in RUN Mode	14
THE SERIAL LOADER & DEMONSTRATOR	15
Description	15
What is the “Smart Autobauding Loader/Demonstrator”?	15
Tips & Tricks	15
Automate Your Testing with ‘Script’ Files	15
Supported Baud Rates	16
Starting and Connecting to the Loader/Demonstrator	16
Communicating with the Serial Loader	17
Command Syntax	17
Addresses and Extended Addresses	17
Special Characters	18
Hex File Record Types Supported	18
Serial Loader Command Reference	19
? - loader on-line help	19
L - Load Intel HEX file	19
D{I R} - dump internal data or SFRs	19
DX [start [end]] - dump intel HEX file	20
V - verify Intel HEX file	20
T - toggle Intel HEX file echo	20

I - toggle interrupt test	20
C [start [end]] - calculate CRC-16	20
E - erase xdata to 0FFH	20
F value [start [end]] - fill xdata with value	20
P value - set default page	20
W {I R} address value - write value to address in idata or SFR space	21
WP val port - put value to microcontroller port	21
WX address value - write the byte to the address in xdata space	21
R {I R} address - read at address in idata or SFR space	21
RP port - get microcontroller port value and display it	21
RX address - read the byte from the address in xdata space	21
M - exhaustive memory test	21
A - address line test	22
X {value} - change movx stretch cycle value	22
WD - write value to DAC (HSM/550 only)	22
S - signature	22
Other new features	22
HARDWARE	23
Controller or ICE Pod Installation	23
Power Supply	23
I/O Mapping	24
External Memory or Peripheral Devices	26
Recommended Peripheral Addressing	26
Protecting Processor Pins from Static or Under-voltage	26
Testing After Adding Peripherals	27
Voltage Monitor, Reset and NVRAM Control	27
Maxcap	27
Lithium Battery	27
Voltage Monitors	28
PFW Interrupt and System Shutdown Time	28
PFW Interrupt Routine Tips	28
Interrupts and Timer/Counter Inputs	29
External Interrupt Pins	29
Interrupts used by UCAN2 peripherals	29
INT0/P3.2	29
INT1/P3.3	29
INT2/3/4/5	30
Dallas MicroLan/iButton	30
DS1820 Temperature Sensor	30
DS1284 Clock and Calendar	31
SBX Mezzanine Connector	31
Systronix SBX1 LCD and Keypad Board	31
Systronix SBX-P Prototyping Board	31
Using Other SBX Boards	31
Using and Modifying UCAN2 I/O and the Prototype Area	32
LED Test Points	32
Adding other Serial I/O	32
S3 and S4 Pushbuttons	32

VCC and GND in Prototype Area	32
DALLAS HIGH SPEED MICROCONTROLLERS	35
High Speed Microcontroller Data Sheets	35
What's Different About the HSM Family	35
Port 0	35
Memory Timing	35
Strobes	36
Instruction Timing	36
Power Supply and Reset Circuitry	36
Additional Features	37
TROUBLESHOOTING & DEVELOPMENT TIPS	38
No Serial Communication between PC and uCAN2	38
Internet FAQ	39
Start Simple	39
Learning Assembly Code and Embedded Programming	40
Exception Handling	40
Quick Diagnosis Table	41
Warranty	42
uCAN2 SCHEMATICS	43

Data Sheet

(Replace with double sided data sheet, PageMaker file)

This data sheet is available in a separate PDF file in the Web/PDF version of this manual.

(back of data sheet)

PLEASE READ THIS FIRST

If you really hate reading manuals, then go ahead and hook up the board - μ CAN2 COM1 is used for program loading (it's marked on the board silkscreen). Use a *straight-thru* serial cable (*not a null modem* cable). Run a simple terminal communications program such as Hyperterminal, set for 19200 8/N/1, XON-XOFF. Send μ CAN2 a carriage return (ODH), usually by pressing the Enter key. μ CAN2 should respond with a logon screen. If you get stuck, come back here and/or check the troubleshooting section or the FAQ at www.systronix.com.

Thank you for purchasing the **High Speed Microcontroller (HSM) Dallas DS80C390 Development System!** μ CAN2 was developed in conjunction with Dallas Semiconductor and customers like you to be the best DS80C390 development board available. The current revision of the circuit board is "A" (in the lower right top corner of the board). μ CAN2 is specifically designed to support the special I/O features of the DS80C390.

■ ***Errata & Anomalies***

UCAN2 PCB Rev A

■ ***Terminology in This Manual***

We use a fixed pitch font to represent what you see or type on your PC:

```
C:\uCAN2\ASM BLINK
```



We use the pointing hand to call attention to something worthy of special note, such as a common pitfall or interesting feature of UCAN2.

■ What You Need to Use uCAN2

- ✓ To power up the UCAN2 board, you need a 6-12 VDC unregulated source such as the Systronix #5003 6VDC 800mA power cube or the heavy-duty #5007 12VDC 1000 mA power cube. If you use your own cube, be sure the center terminal of the 5.5 x 2.5 mm jack is positive 6 to 12 volts DC. The sleeve is negative. An AC power cube could damage UCAN2.

To connect uCAN2 to your PC serial port, you need a **straight-through cable from your PC** to a DB9 female, to mate with the DB9 male on UCAN2. Systronix #9210 serial adapter kit contains a 6 foot/ 2 meter DB9 extension cable, a DB9 to DB25 adapter, and DB25 and DB9 gender changers. **DO NOT USE A NULL MODEM CABLE. COM1 of uCAN2 is the loader serial port.**

- ✓ To download a program to uCAN2, and communicate with the uCAN2 serial loader, you need any serial communications program such as Windows Hyperterminal. You can use *any* computer, not just a PC-compatible, as long as it supports RS232 communication.
- ✓ To assemble a program for use on μ CAN2, you need an 8051 assembler such as the Systronix RAD51 assembler included with mCAN2 Or you can use any 8051 family development tool, running on any computer platform of your choice, as long as it generates a standard Intel HEX file. Binary file loading is not supported by mCAN2.
- ✓ To understand the High Speed Microcontroller family, you need the Dallas High Speed Microcontroller data sheets and application notes, available from www.systronix.com, www.dalsemi.com, or by calling Dallas Semiconductor at 972-371-4000. We've included the DS80C390 data sheet in PDF format on your mCAN2 disk.

QUICK START - STEP BY STEP

■ **What you need**

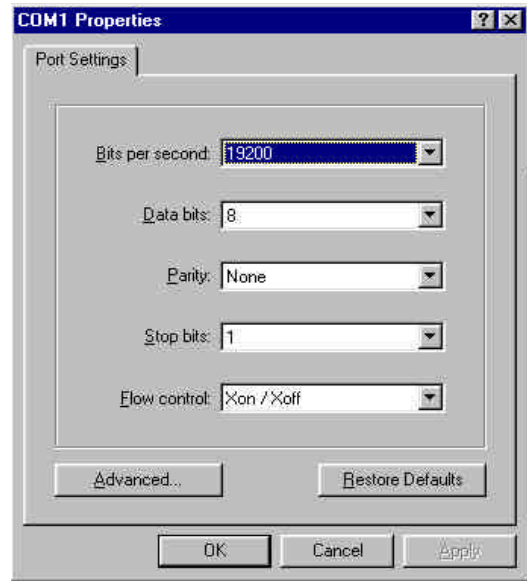
If you've lost any of the accessories provided with μ CAN2, contact us at 801-534-1017 or email to support@systronix.com.

- 1) μ CAN2 board, with the power cube (included with all orders)
- 2) Serial cable to your PC (you provide this or order our serial adapter kit) and terminal software such as HyperTerminal (you provide this).
- 3) Systronix RAD51 assembler (provided on mCAN2 disk or available at www.systronix.com), or an 8051 C Compiler such as Keil C.
- 4) Text editor which can save ASCII text. Windows Notepad or Wordpad are OK.
- 5) Clip leads to connect the LED test point to a processor port pin. Clip leads with a .025 square receptacle or a micro hook on each end are ideal. We'd like to include these with μ CAN2 if we can find a good source at a reasonable price.
- 6) Jumpers to connect μ CAN2 interrupts to the pushbuttons (provided with mCAN2)
- 7) DS80C390 data sheet (provided on mCAN2 disk or available at www.systronix.com or www.dalsemi.com)

■ **Step-by-Step Procedure**

- 1) μ CAN2 example files install automatically as part of the CD-ROM self-installer. They will be located in a CODE folder which is a sub-folder of your main installation folder. By the way, the folder is named ' μ CAN2', not ' μ CAN2' since some programs have trouble with the Greek mu character as part of a file name.
- 2) Connect the power cube to the P17 power jack. After application of power, the green RUN LED will be lit. Press the LOAD switch and hold it for more than 300 msec. The red LOAD LED should be on. If neither LED is lit, the board is not receiving power. Check for proper power polarity - the center of the P17 power jack is positive.
- 3) Connect a DB9 female straight (*not null modem*) cable from your PC serial port to COM1 (near the prototype area). mCAN2 expects TXD on pin 2 and RXD on pin 3 of the serial port cable. This sets up the PC as "DTE" (Data Terminal Equipment) and μ CAN2 as "DCE" (Data Communications Equipment) in RS232 parlance. A kit of adapters and a DB9 extension cable is available from Systronix at 801-534-1017 or www.systronix.com.

uCAN2 COM1 is the added UART of the Dallas controller, and is used by the serial loader for program loading. uCAN2 COM0 is the normal UART available in most every 8051, and is not used at all by the serial loader. mCAN2 must use COM1 for loading but you can use any available PC COM port.



- 4) Start a terminal program such as Hyperterminal, set the communication parameters for the baud rate you desire (typically 19.2 kbaud). Set other communication parameters to “direct” connection (not through a modem), 8 data bits, no parity, 1 stop bit, Xon/Xoff flow control. A Hyperterminal dialog box is shown to the right. Be sure to set the terminal software COM port to the correct port on your PC. Save these settings. Now open a connection in the terminal software, usually with a “connect” command or button.

- 5) Wait a second, to give the terminal program time to establish the connection to mCAN2. Now press and hold the LOAD button for more than half a second. When you release the button, the red LED should remain lit. If, instead, the green LED lights, you need to hold the LOAD button longer before releasing it. Now press your PC’s ENTER key to give the loader a carriage return to use as its auto baud rate character.

 When you first receive the board, program memory will be empty.

If your PC serial port is connected through a straight cable (TXD and RXD are not swapped), and you press the LOAD button on uCAN2, and then press the ENTER key on your PC, you should see the loader prompt. It will look something like this. The exact appearance may vary with your specific loader version.

```
Unified HSM HEX LOADER Rev E.00 (8/01/99)
(C)1996-1999 SYSTRONIX, INC.
Type "?" for command help
```

- 6) If you enter a question mark for help, you should see something like this (the exact appearance will depend on your loader version):

```
uCAN2> ?
----- Loader Commands -----
L                               ;load intel HEX file
D [start [end]]                ;dump HEX file
V                               ;verify HEX file w/memory
T                               ;toggle HEX file echo
I                               ;toggle all external interrupt enables
C [start [end]]                ;calc CRC-16
E                               ;erase xdata to 0FFH
F value [start [end]]          ;fill xdata with value
G                               ;get port values
P val0 val1 val2 val3          ;put valx to portx...
                               ;...except TXD1, RXD1, RD, WR of P1 & P3
WX adr val                     ;write val to adr in xdata or I/O space
WI adr val                     ;write val to adr in internal data space
WR adr val                     ;write val to adr in SFR space
RX adr                         ;read at adr in xdata or I/O space
RI adr                         ;read at adr in internal data space
RR adr                         ;read at adr in SFR space
M                               ;Memory test with exhaustive patterns
A                               ;Address line test
X value                         ;show/change movx stretch value (0-7)
-----

Press any key to continue . . .
  WD val                       ;write val to system DAC (550 only)


Memory: code=0000-EFFFH, xdata=0000-EFFFH, I/O=F000-FFFFH
All values are entered and displayed in hexadecimal format
memory errors display as {address}:{should be}/{actual}

Loader has detected DS80C390 processor
uCAN2>
```

- 7) Try sending one of the HEX files we've included to uCAN2. To send a HEX file to μ CAN2, type an "L" for LOAD, and send the HEX file from your terminal software. If the loader receives the file with correct checksums, you will get an "OK" response. If there are load errors, the loader will tell you as they occur.

It's tempting to jump into a complex application right off the bat, but *please run one of the supplied sample programs first*. This will verify that your PC, cable, hardware and installed software are all working together

If you are using assembly code, we've included example files "HELLO33.HEX", "DS1820.HEX" and others, along with the .ASM source code. Check our web site www.systronix.com for other samples. For a quick test of your system, load the HELLO40.HEX file. This program continuously prints "hello from COMX" to both UARTs assuming a 40 MHz system speed. HELLO20 and HELLO10 are versions for 20 and 10 MHz, respectively.

 There are many differences between the DS80C390 and other 8051s.

- 8) To run your loaded file, move the serial cable to COM0 (the default “user” serial port), and press the RESET button. The green RUN LED should light, and your sample program will emit a repeating message to your terminal software screen. Since the serial loader always uses COM1 and generic 8051 code will always use COM0 for serial I/O, you can leave both serial ports connected - COM1 to your PC, to load the program to be tested, and COM0 to another port on your PC, a different PC, serial modem, or other serial device which you are controlling.

- 9) That’s it! You should be up and running. If you had problems, check your PC serial port connection and refer to the troubleshooting portion of this technical reference.

DETAILED DESCRIPTION

■ **Philosophy and Purpose of uCAN2**

uCAN2 is the first development system for the Dallas DS80C390 CMOS Dual CAN High-Speed 8-bit Processor. Like our HSM/550 and other boards, uCAN2 is designed from the ground specifically for the Dallas High Speed Microcontroller (HSM) family. There are some differences between generic 8051s and the HSMs. Just plopping an HSM into an 8051 socket will probably not give you the best performance, and will probably not even work at 22 Mhz or above. These differences are detailed in a separate section of this manual.

The power supply is a low-noise, high-efficiency switching regulator. The switcher circuit is a proprietary Systronix design which has very low output impedance and very low ripple. It is actually quieter than many analog regulators!

- a. Easy to use eval/development board for the Dallas DS80C390.
- b. *Full featured* - complete with 1 MByte each of 40 MHz code and data memory (we sell no boards with empty sockets or slow memory as some vendors do to advertise a low price).
- c. Clock and calendar for time stamping or time-based control.
- d. Designed for prototyping and development - generous through-hole and surface mount prototype areas, clearly labeled access to all controller I/O port pins.
- e. SBX connector for additional I/O or prototyping expansion.
- f. Easy to use - all you need to load a program is a PC with a serial port.
- g. Built in auto-bauding serial loader for easy program loading and testing.
- h. High Speed - designed for use at up to 40 MHz with no data “stretch” cycles. In order to run at full speed, the Dallas HSMs have special needs compared to generic 16 MHz 8051s. uCAN2 has been designed specifically for the HSM family. “Accept no substitutes”.

■ **Board Versions**

uCAN2 is available only in a 40 MHz version. You can select an execution speed of 40, 20, or 10 MHz in your firmware. You can change to a slower crystal if you wish. Be sure to use a parallel resonant crystal designed for 18-20 pF load capacitors. Systronix stocks such crystals in 33.0000, 22.1184, 14.7456, 11.0592, 10.0000 and 20.0000 MHz versions.

Check our web site for special offers or special bundled versions. We also offer aggressive quantity and educational discounts.

■ Features

- a. PLCC68 socket for DS80C390 High Speed Microcontroller or In Circuit Emulator.
- b. 2 MBytes 20 nsec SRAM with MaxCap backup, divided into code and data pages. On board circuitry protects the SRAM from invalid write cycles during power up and power failure conditions.
- c. Designed to operate at the maximum controller speed of 40 MHz with no wait states or stretch cycles. This requires VERY fast SRAM for both program and data memory. Readily available EPROM and Flash are far too slow. In August 1999, the fastest available 128K or larger EPROMs are 45 nsec, and Flash is 70 or maybe 55 nsec. The 20 nsec SRAMs used on uCAN2 are relatively power hungry and expensive. We included enough memory to contain a large program and a significant amount of data. The design of uCAN2 is optimized for speed, it is not intended to be a low-power or low-cost board.
- d. 4 Kbytes of memory-mapped I/O space.
- e. Dual CAN 2.0B channels with twisted pair cable drivers.
- f. On-board auto-bauding serial loader accepts a HEX file from UART1. Leaves UART0 untouched. Use any terminal program to transfer the file. Does not use DTR to trigger load mode.
- g. Serial I/O: RS232 on UART0 and UART1.
- h. Early power fail interrupt provides time for orderly system shutdown.
- i. On-board high efficiency, low noise switching regulator, 6-12 VDC input, 5.5x2.5 mm power jack. The regulator can provide up to 500??? mA at 5 volts for your use, provided that you use a suitable input power source for the board.
- j. Push-button high drives one or more of INT2, INT3, INT4 OR INT5 (jumpers select).
- k. Push-button low drives one or more of INT0 or INT1 (jumpers select).
- l. LEDs to indicate load (red) or run (green) status.
- m. Two amber LED test points - LED lights when point is driven low. Requires less than 1 mA of low sink capability on the driving source, so can be driven by any port pin.
- n. Microcontroller supervisor and reset circuitry, with early Power Fail Warning interrupt to INT0 (can be isolated if you don't want PFW driving INTO).
- o. Generous prototyping area for DIPs and also an *SMT proto area!* Proto area has a power plane on the front side of the board and a ground plane on the back. Multiple solder pads for power on the front and ground on the back make connecting your own chips quick and easy.
- p. All processor signals brought out to headers, labeled on *both sides of the board* for easy wiring and probing.
- q. Serial loader accepts Intel Hex files from any terminal communication program. The serial loader is active only in LOAD mode or when first booting up. In RUN mode it is mapped out of processor memory and does not use any processor resources. Our serial loader is completely non-intrusive.
- r. Dallas MicroLan connector with on-board DS1820 temperature sensor.
- s. On-board DS1284 clock and calendar with battery backup. The DS1284 can generate

- periodic- or calendar- based interrupts.
- t. Standard Euroboard 160x100 mm size, conforms to ANSI/IEEE Std 1101/1987, IEC 297-3-1984, and other international standards. uCAN2 fits Systronix enclosures and many others.

■ *Loader and PLD Options and Updates*

We are planning to add additional features in future releases, and we welcome your input. Please contact us if you would like a special version of either the loader or any of the PLDs. We can design and produce hardware built to your specifications, and have done so for a variety of small and large companies.

If you wish to use the loader or PLD in your own products, we can provide programmed parts or a license to create your own, at a very reasonable price. We have, in fact, done so for several customers. The serial loader and memory control PLD, and their associated circuitry are copyrighted by Systronix, Inc, with all rights reserved. Purchasing uCAN2 does not give you the right to duplicate either device.

■ *Systronix Web Site & Forum*

Our web site (www.systronix.com) is the main repository for new uCAN2 example code and documentation. You can also join a forum of Systronix users. Our web site has information about the forum..

■ *Getting Technical Support*

Our technical support is included with your purchase of uCAN2. We believe support begins with good written documentation (starting with what you're reading right now). If you can't find the answer in our documentation, then try the FAQ on the web at www.systronix.com, send e-mail to support@systronix.com, call us at 801-534-1017, or Fax us at 801-534-1019. When you contact us, please tell us about any errors or weaknesses in the documentation so that we can improve it in the next revision.

If you can, please contact us by e-mail first. You can attach a file of source code and captured output (use MIME encoding if possible) to your message. If you send us an example of a problem please make the example as simple as possible, and include any necessary I/O driver "include" files if you have modified them. *Please send us ASCII text or PDF files rather than word processor files.* Due to the risk of viruses, we cannot accept word processor files containing macros. We try to answer all e-mail within one or two business days.

Support does not include designing or writing code for you or debugging your application. We'll be happy to give you some general suggestions. Beyond that, if you need an application developed, we offer that service as well.

Please feel free to contact us with any unusual questions about programming uCAN2. We

can probably help you approach your needs in the most efficient way. That's why we're here! Customers consistently give us high marks for courteous, competent technical support. But, we're only human and sometimes we make mistakes. We listen to our customers (we've gotten some of our best product ideas from them!). If you're happy with our service, please tell a fellow developer. If not, please tell us so that we can make things right.

■ *Installing and Using the Windows RAD51 IDE & Assembler*

RAD51 requires 32-bit Windows such as Windows 95, 98 or NT 4.0 or later. It is not compatible with Windows 3.X and has not been tested with older version of NT. RAD51 installs with it's own self-installer program which auto-launches when you insert the CD-ROM into a drive. Online documentation for the assembler is included.

uCAN2 MEMORY

uCAN2 Memory Map in Load and Run Mode

The table *uCAN2 Serial Loader Modes and NVRAM Memory Access* describes the relationship between load and run modes and NVRAM page access.

Memory Backup

uCAN2 uses four 20 nsec 512Kx8 SRAMs with nonvolatile control logic and a MaxCap for program retention. The SRAMs are power hungry and even in standby mode can consume 10 mA each. Yes, that's milliamps, not microamps. This makes battery backup impractical. I²C and EPROM are both too slow to operate at 40 MHz. (This is the same reason Pentium-class PCs have special dynamic RAM designs and include fast SRAM cache). These factors drive the design and make the manner in which uCAN2 operates a little different from all our other development boards.

The backup time of the memory is 45 seconds (calculated worst case) to 2 minutes (typical observed at room temperature). This is long enough to span typical AC brownouts, and to permit cycling the board power while you attach test leads, scope probes, etc. If power has been off any length of time, the program will have faded from memory. It's important that a board without a valid program not be allowed to boot up and attempt to execute whatever it finds in memory. This might be controlling a machine tool, motors or other devices which must always be kept in a predictable and controlled state.

Nonvolatile System Parameters and Unique Network ID

uCAN2 includes an Embedded Clock and Calendar (ECC) with it's own Lithium battery (typically good for 10 years). The ECC device has several bytes of available NVRAM. This makes a handy place to store some system flags. uCAN2 also includes a Dallas DS1820 temperature sensor with a unique serial number. These two devices then give each uCAN2 a unique identifier and a means of safely storing system parameters over extended periods of no external power.

Load Mode

After a power on or pushbutton reset, uCAN2 boots up in the loader. The loader checks P3.5, the RS232 shutdown and load pushbutton pin. If uCAN2 is supposed to stay in load mode (because the load pushbutton is pressed) then the loader waits for a carriage return and operates as a loader.

While executing out of the loader EPROM, the loader can still access data bank NVRAM as data. This is different from how the loader works on the HSM320, 520, 530, and 550 boards, in which data NVRAM is not accessible to the loader.

All 8051's must boot up by executing code starting at address 0H, the so-called boot or startup vector. This is where the loader EPROM must be addressed. In order to write to 'code' pages, the loader must temporarily map the 'code' memory as data, starting at address 0H of page 0. Of course, the loader itself must be executing out of code space. But the code space SRAM cannot be mapped to address 0 at the same time the loader EPROM is also mapped there. On the C390 this is resolved by copying a small section of program loading code into the special C390 internal SRAM and executing it there. This special internal 'code' SRAM is unique to the C390. The loader spends as little time as possible executing from internal SRAM, and does not handle interrupts there.

RUN mode

If the load pushbutton is not detected, then the loader copies a CRC check routine into the C390 internal SRAM. The loader sets the memory map bits so that code NVRAM is accessible as data. The loader must do this because it is running in low code space, and can't also access the user NVRAM code space at the same addresses. Execution branches to the loader routine which is in the C390 internal SRAM. There memory control bits are set which make code reachable as data. The code fragment then checks the CRC of the first 64K page of user program memory. It's CRC should match the one stored in the clock and calendar at addresses E and F.

If the CRC test passes, then the loader assumes a valid user program exists in NVRAM code space. The loader again copies a code snippet into high C390 SRAM and execution branches there. That code changes the memory map to address code NVRAM as code, and does a firmware reboot at location 0 in the code NVRAM. Finally, then, your program has been verified and its execution has begun.

uCAN2 Serial Loader Modes and NVRAM Memory Access		
Values are Hexadecimal with format PP:AAAA where PP is the upper page value and AAAA is the 16-bit location within that page. (Decimal values are in parentheses)		
Reset Type:	Code Space	Data Space
Power-On Reset - or - pushbutton RESET without LOAD button held down - or - RUN mode	CE0: 512Kx8 Code page 0 0-7FFFF (0-512K)	PCE0: 512Kx Data page 0 0-7FFFF (0-512K)
	CE1: 512Kx8 Code page 1 80000-FFFFFF (512K- 1024K)	PCE1: 512Kx8 Data page 1 80000-FFFFFF (512K- 1024K)
	CE2: unused	PCE2: 512Kx8 memory mapped I/O space (only 4K used) 10:0000-10:0EFF(1048576- 1052671), 10:0FXX-10:0FFF is reserved. note 1
	CE3: unused	PCE3: defined as I/O bit, we use it as an EPROM enable control pin
		1Kx8 SRAM (Data or stack) 400000-4003FF (4194304- 4195327)
		3Kx8 SRAM (Data) 400400-400FFF
		256x8 CAN0 Data 40:1000-40:10FF
	256x8 CAN1 Data 40:1100-40:11FF	
LOAD mode (LOAD button held down for approximately 1 second) Loader <u>not</u> accessing NVRAM code pages	CE0: 32Kx8 Serial Loader EPROM 0-7FFF, rest unused	PCE0: 512Kx8 Data bank 0 0-7FFFF (0-512K)
	CE1: 512Kx8 Code Page 1	PCE1: 512Kx8 Data bank 1 80000-FFFFFF (512K- 1024K)
	CE2: unused	PCE2: memory mapped I/O space
	CE3: unused	PCE3: defined as I/O bit, we use it as an EPROM enable control pin
		internal SRAM same as in RUN mode, including CAN space

LOAD mode, loader is accessing NVRAM code pages	CE0: accessing code bank 0 in special write mode, movx reads activate PSEN for read and movx writes use WR for write Note 2	PCE0: disabled, no data space access possible when reading or writing to NVRAM code banks
	CE1: accessing code bank 1 in special write mode same as CE0	PCE1: disabled, no data space access possible when reading or writing to NVRAM code banks
	CE2: unused	PCE2: memory mapped I/O space
	CE3: unused	4Kx8 SRAM (used as loader code space) 400000-400FFF (4194304-???)
		256x8 CAN0 Data 40:1000-40:10FF
		256x8 CAN1 Data 40:1100-40:11FF
		PCE3: defined as I/O bit, we use it as an EPROM enable control pin

Note 1: memory mapped I/O space only decodes the lower bits of the 512K address block assigned to PCE2. Therefore it “reappears” at every 4K boundary of the I/O space address range.

Note 2: The loader EPROM and the code page SRAM cannot exist at the same code space address at the same time. So we copy a section of loader code into the internal code/data SRAM and execute there.

Serial Loader 100% Non-intrusive in RUN Mode

The serial loader is active only in LOAD mode. Our proprietary loader and hardware design provides this function. In RUN mode, the serial loader does not use *any* processor resources. **In RUN mode our serial loader is completely non-intrusive and will not interfere with your application program in any way.** This is a very important capability, and one of the nicest features of this system.

The serial loader includes other functions such as memory test, reading and writing to controller ports, and so forth. It is not a monitor, so it cannot be accessed from your application program, and cannot set breakpoints, single-step your program, and other typical monitor functions. Monitors which do this should be part of your executable program. Such monitors are available from a number of sources, some free. If we ever get a lot of time on our hands we may develop one ourselves.

THE SERIAL LOADER & DEMONSTRATOR

■ ***Description***

What is the “Smart Autobauding Loader/Demonstrator”?

That’s a rather ungainly name, but it was the best we could think of. (We’re not an ad agency, just a bunch of acronym- and technology- loving engineers). The loader provides Intel HEX file transfer, memory fill, HSM Port read and write, SFR access, I/O space peek and poke, and much more. I/O space “peek and poke” is really just I/O space “read and write” but “peek and poke” sounds like more fun, doesn’t it? *You can use any terminal emulator or communications program which supports standard RS232 serial I/O.* You don’t need a “Wintel” PC.

Tips & Tricks

Automate Your Testing with ‘Script’ Files

In load mode, you can of course send a series of keyboard commands to the loader. This could include erasing memory, writing and reading memory-mapped I/O space, sending a HEX file and so forth. “OK”, you say, “maybe that’s cool but where’s the tricky part?”

Here it is: you can prepare an ASCII text file with these commands, one per line. Then send the file to HSM. In Hyperterminal it’s Transfer->Send Text File. You will need to set the File->Properties->Settings->ASCII Setup->Line Delay to 500 or 1000 msec, to give the loader time to process the command. If your communications software supports macros or scripts, so much the better. We’ve included some sample files we use with Hyperterminal. More are on the web site.

This is a simple and powerful way to test register settings before you take the time to create a program. It’s much easier to debug a configuration problem at the loader command line, than within a program. We use this approach a lot in our own test and debug work here. That’s why we took the trouble to add these features to the loader - we thought you’d find them useful too.

Supported Baud Rates

The Loader/Demonstrator syncs to many standard baud rates from 600 to 57,600 baud depending on the controller crystal. Note that with a 33 MHz crystal, 38,400 baud error is greater than 3% and may not be reliable. Crystals on the HSM board and in your PC are not

perfect, especially at temperature extremes, so you may find that your PC does better or worse than the table indicates. For example, one new NT workstation here at Systronix World Headquarters (WHQ) does fine with a 33 MHz HSM at 38400 baud, while another identical workstation occasionally receives garbled characters.

HSM Loader/Demonstrator Baud Rates								
Crystal	600	1200	2400	4800	9600	19200	38400	57600
33.000 MHz	Y	Y	Y	Y	Y	Y	error >3%	Y
22.1184 MHz	300	Y	Y	Y	Y	Y	Y	N
14.7456 MHz	150	Y	Y	Y	Y	Y	?	N
11.0592 MHz	150	Y	Y	Y	Y	Y	N	N
10.000 MHz	Y	Y	Y	Y	Y	Y	Y	N
20.000 MHz	Y	Y	Y	Y	Y	Y	Y	N
40.000 MHz	Y	Y	Y	Y	Y	Y	Y	Y

Starting and Connecting to the Loader/Demonstrator

The HSM serial loader/demonstrator uses the HSM second serial port (COM1 on the HSM board). If your board uses an external EPROM for the loader, it typically resides in a 27C256 EPROM or equivalent. Your board's ROM select jumper must be in the "EXT" position to drive the HSM's EA (External Access) pin low. Normally, we ship all boards with this jumper in the EXT position (so we can test the loader). If your board has the loader in internal EPROM, then the ROM select jumper must be in the INT position.

Connect your computer or terminal to the development board with the appropriate cable, connected to the correct development board serial port. Set up your PC communications for RS232, XON-XOFF, 8/N/1, and 19200 baud (you can increase the baud rate later). Put your board into LOAD mode (usually by pressing and holding the LOAD pushbutton for a second), and you will see a sign-on message on your PC similar to this:

```
Unified HSM HEX LOADER Rev D.04 (08/31/99)
(C)1996-1999 SYSTRONIX, INC.
Type "?" for command help
```

If you enter a question mark for help, you should see something like this (the exact appearance will depend on your loader version). (Loader commands are described in detail later in this section.)

```
?
----- Loader Commands -----
```

```
L                ;load intel HEX file
D{I|R}          ;display content of idata or SFRs
DX [start [end]] ;display content of xdata
V              ;verify HEX file w/memory
T              ;toggle HEX file echo
I              ;toggle all external interrupt enables
C [start [end]] ;calc CRC-16
E              ;erase xdata to 0FFH
F value [start [end]] ;fill xdata with value
W{I|R} adr val  ;write val to adr in idata or SFR space
WP port val    ;write val to port
WX adr val     ;write val to adr in xdata or I/O space
R{I|R} adr     ;read at adr in idata or SFR space
RP port        ;read port
RX adr         ;read at adr in xdata or I/O space
M              ;Memory test with exhaustive patterns
A              ;Address line test
X value        ;show/change movx stretch value (0-7)
```

Memory: code=0000-EFFFH, xdata=0000-EFFFH, I/O=F000-FFFFH
All values are entered and displayed in hexadecimal format
memory errors display as {address}->{should be}/{actual}

■ **Communicating with the Serial Loader**

Command Syntax

All loader values (addresses, data, and all parameters) must be provided in hexadecimal format, without any special characters. For example, 0ff, ff, FF, and 0FF are valid values. 0xff and 0FFH will be considered errors by the loader.

All commands are single characters with optional arguments. Arguments must be separated by one or more tab or space characters. The entire command line must be completed with a carriage return.

D{I|R}

means that commands DI and DR are both legal.

F value [start [end]]

Parameters not in square brackets are required. Parameters in square brackets are optional. In the example above, the Fill xdata command, the fill value must be provided. Start and end memory locations are optional.

Addresses and Extended Addresses

Addresses given to the loader are interpreted in light of the target controller and its development board. Some boards support extended addressing - i.e., paged memory. Boards which do not support extended addressing will ignore any extended address records in Hex files, and emit an error message if you manually enter a page address as a command parameter.

To save retyping the page value in extended addresses, there are some shorthand techniques you can use. Addresses are entered as

```
page:address
```

where a value followed by a colon is assumed to be the page. If not followed by an address, then the entire page is assumed, starting with location 0. For example,

```
page:
```

is equivalent to

```
page:0000
```

Some examples:

```
F 55 05:
```

fills page 05 from 0000 through FFFF (excluding any I/O space) with the value 55.

```
DX 0:100 2:200
```

will dump xdata from page 0, address 100 through page 2, address 200. The page value is not needed if you are assuming the current default page, so if the current page is page 0, then the above DX command is equivalent to:

```
DX 100 2:200
```

Special Characters

Control-C (^C, 03 hex) will restart the loader. A backspace key (08 hex) causes a destructive backspace. The loader always uses xon (^Q - 11 hex) and xoff (^S - 13 hex) flow control. The flow control is accepted as input from the host PC - in other words the PC can pause the loader's output. The loader does not emit flow control characters to the PC, so the PC can send a stream of data at the selected baud rate and the loader is guaranteed to keep up. As you use a slower and slower crystal, the fastest baud rate supported also decreases (9600 baud max at 1.8432 MHz for example), so the maximum incoming data rate is self-limiting.

The loader uses XON-XOFF to emit warning or error messages during hex load operations.

Hex File Record Types Supported

Hex file record type 00 is a data record. Record type 01 is end of file. These both occur in every typical Hex file which you will generate with any standard 8051 assembler or compiler.

Hex file record type 02 is an extended address record, used to set the high address byte (either the page value or the upper byte of the address on controllers such as the DS80C390). On boards which do not support extended addressing, this record type is ignored. A warning is emitted but the HEX file continues to load. Subsequent data records can overwrite previous data. It's up to you to be sure your HEX file is correct, the loader won't prevent you from doing something unusual.

Hex file record type 03 is a start address record. Some compilers (BSO Tasking for example) generate this record. It's intended to be used to relocate code, perhaps in an EPROM programmer. It's not clear to us what purpose this could have in an 8051 system loader, so the loader ignores it.

■ *Serial Loader Command Reference*

? - loader on-line help

This command causes the loader to emit several lines of command help.

L - Load Intel HEX file

This command tells the loader to await reception of an Intel HEX file, and upon its reception, to load it into what will be the user code memory. On some targets, code memory is mapped as data in LOAD mode, in order to make it writable. In RUN mode, the successfully loaded Hex file should execute as code.

While loading, the code byte is written and then immediately read. Errors are reported as they occur. The incoming file is not echoed to the serial port unless the T command has been given prior to the L command. If the load is error-free, the loader will issue an "OK" response. The loader recognizes x-on/x-off flow control coming from the Hex file sender (i.e. your sending PC can pace the output from the loader). Extended address records are ignored if the target is a development board which does not support paged memory.

Give the loader the L command, then send a HEX file as standard ASCII text - do not use protocols such as zmodem. In HyperTerminal this means using the "Transfer->Send Text File" menus. To be sure your entire Hex file was loaded correctly and that portions of it did not overwrite itself, use the Verify command after loading.

D{I|R} - dump internal data or SFRs

Send the contents of idata or SFR space to the serial port, in HEX file format. Only useful if you are familiar with these portions of the controller. These are not generally useful as debugging aids since the loader uses idata and the SFRs, so their contents may have changed from the time your application program was running.

DX [start [end]] - dump intel HEX file

Send the contents of xdata space to the serial port, in HEX file format, beginning at address {start} and proceeding through address {end}. If no addresses are provided, dumps the current page. On systems which map code into data in LOAD mode, DX is actually dumping what will become code space in RUN mode.

V - verify Intel HEX file

Tells the loader to await reception of an Intel HEX file to be compared to those addresses in the development board's memory. After you enter the V command, send a HEX file just as if you were using the load command.

T - toggle Intel HEX file echo

Causes the loader to echo all incoming HEX files back out the serial port so that you can see it as it's being received. The echo persists until you enter the T command again or reset the loader.

I - toggle interrupt test

This is not just a loader command, it is also interrupt vector code contained within the loader. If you trigger an external interrupt while the loader is active, and interrupt test is enabled, the loader will emit a message to HSM's COM1. This is useful for verifying that your interrupt hardware is working correctly, or for testing the pushbuttons on your board. All interrupts are assigned the same priority level. This feature is only available in LOAD mode and does not in any way affect your program's interrupts in RUN mode.



An interesting test is to jumper-enable multiple interrupts. When you press the button, all jumpered interrupts will be asserted at the exact same time. This is a good test case for interrupt handling code. Serial loader interrupt code handles each interrupt in order of its priority. No interrupts are lost. Try it! Then change the interrupt priority with an appropriate WR command and try it again.

C [start [end]] - calculate CRC-16

Calculate a 16-bit Cyclic Redundancy Code. If no addresses are provided, calculates over the default page. The addresses can span multiple pages.

E - erase xdata to 0FFH

Erase all of xdata to the value FF, except for I/O space. On systems with multiple memory pages, all pages will be erased.

F value [start [end]] - fill xdata with value

Fill all of xdata with the value provided, except for I/O space. If no addresses are provided, fills just the default page. (The default page value is displayed in the loader prompt.)

P value - set default page

Available only on systems with extended addressing.

W{I|R} address value - write value to address in idata or SFR space


Write the value to idata or SFR space. You must use the SFR address, not its assembly code label. For example, TMOD is at address 89H. The WR command lets you write any Special Function Register. Timed Access Registers are supported, and the loader performs that special access for you. There are no restrictions, so you can, if you wish, clobber the serial I/O used to access the loader. If this happens just reset the board.

WP val port - put value to microcontroller port

Puts the value you provide to the port, except for port pins which are used for TXD1, RXD1, RD, WR of Port1 & Port3.

WX address value - write the byte to the address in xdata space

This command lets you set individual bytes of xdata space, and write to any location in memory-mapped I/O space. Xdata space typically includes I/O space at F000 and above. On many development systems, xdata space in LOAD mode becomes code space in RUN mode.

 This command gives you the ability to manually write to any peripherals you've installed in the prototype area as memory-mapped I/O. This is very useful in debugging your hardware.

R{I|R} address - read at address in idata or SFR space


Read the value in idata or SFR space at the given address. You must use the SFR address, not its assembly code label. For example, RR 89H reads the TMOD SFR. Timed Access Registers are supported, and the loader performs that special access for you.

RP port - get microcontroller port value and display it

If you use the board's pushbuttons to drive some of the Interrupt pins, you should see the change in values with the RP command. For example, HSM/320 INT0 (P3.2) driven low causes RP 3 to return FB, undriven, port 3 is FF.

RX address - read the byte from the address in xdata space

This command lets you read individual bytes of xdata space, including any location in I/O space.

 This command gives you the ability to manually read any peripherals you've installed in the prototype area as memory-mapped I/O. In combination with the W command, you can manually configure and test all your I/O devices.

M - exhaustive memory test

This command does not access I/O space. This command writes a series of patterns to memory and reads them back. Errors are reported as they occur. This tests all bits of memory with ones and zeros, and will detect shorted address or data lines. It doesn't help you find which lines are shorted, however. On many development systems, xdata space in LOAD mode becomes code space in RUN mode - on such systems this command tests what will be code space.

On systems with multiple memory pages, all pages will be tested. This can take a few minutes, during which time the loader emits a series of periods, two per page, to indicate that the test is running.

A - address line test


This command does not access I/O space. This command writes a series of patterns to memory and reads them back at locations calculated by walking a 1 across the value of the memory address. Errors are reported as they occur. This test is good for detecting shorted address lines. For example, if you get errors at locations 1000 and 4000, then A12 and A14 are shorted together.

X {value} - change movx stretch cycle value

This command changes the value of the controller's movx stretch cycle register. The movx stretch cycle is similar to "wait states" in a microprocessor memory cycle. After a reset, the

value defaults to 1. It can be anything from 0 (no stretch cycles) to 7 (for the slowest write cycle).

Any change to the value persists until the next board reset. It applies to all movx cycles, including memory test and I/O access. All our development boards are designed to support a stretch cycle value of 0 in all write accesses to on-board data memory. If you add peripherals to the prototype area, you may need to use a slower write cycle when you access them.

 Connect an oscilloscope to the WR strobe (P3.6, on the Port3 header). Run the A (address line) test or M (memory) test. Adjust the 'scope for a convenient display of pulse width. Now enter the command "X 0" to change the stretch cycles to zero. Run the memory test again, watch the oscilloscope, and you can immediately observe the difference in write pulse width.

WD - write value to DAC (HSM/550 only)

This command is only available on systems which have a DAC, such as the HSM/550. Meaningful values are 0-FFF (12 bits, 0-4095 decimal).

S - signature

If you don't read this manual, you won't learn about this command, since it's not in the on-line loader help. This command emits the signature of the authors of the serial loader. It is located at the top of code memory in the serial loader, so it is a good test of the integrity of the loader upper address lines.

Other new features

See the readme file on disk, or the loader help screen for information on new features which your loader may contain.

HARDWARE

■ **Controller or ICE Pod Installation**

A PLCC68 part has one chamfered corner which must be aligned with the matching socket corner. Device leads are relatively fragile so be careful to line up the part in the socket precisely before pressing it in place. Refer to the schematics and the Dallas data sheets for processor pinout details and signal names.

If you are using an In Circuit Emulator (ICE), turn off the board and ICE power. Insert the ICE pod, then turn on the ICE and board power at the same time. We recommend using a power strip to do this. If you power one and not the other, you could cause CMOS components in the ICE or the board to latch up, possibly destructively. We've seen this erase PLDs and damage the ICE. That's when we started using a power strip to turn both the board and the ICE on and off simultaneously.

■ **Power Supply**

uCAN2 can be powered either from the provided power jack or from the CAN network cable.

The power input jack is 5.5 x 2.5 mm, with center terminal positive and the sleeve negative. The voltage regulator will accept supplies up to an absolute maximum of 24 VDC.. Power supply input voltages above 6 VDC are acceptable, and up to a point (about 9V DC) will make the regulator more efficient. The regulator will simply shut down if it is short-circuited or overheats. The regulator will start to drop out at about 5.5 volts typical. Reset circuitry puts the board in a safe reset state if VCC drops to 4.5 volts.

If you provide your own power supply be sure it is a DC voltage of at least 6 volts. Some wall cube power supplies are actually AC transformers and may damage the board. uCAN2 ships with a 12 VDC 1000 mA power cube. With an input voltage of 12 VDC, uCAN2 draws ??? mA typical, and with 6 VDC input, ??? mA. Current use is lower at higher input voltages because of the power conversion of the switching regulator (Power = Voltage X Current X efficiency_coefficient).

If you will be driving the board from a 12 VDC vehicle electrical system, the input voltage will typically be 13.8 VDC. uCAN includes reverse-polarity and overvoltage protection. The overvoltage will pass 24 VDC but clamps anything above that. ??? How far above

■ I/O Mapping

uCAN2 uses memory-mapped I/O to support on-board memory-mapped peripherals, the SBX connector, and your own prototype area devices. uCAN2 reserves 4 Kbytes of external data space for memory-mapped I/O at address FXXXH -FFFFH, within page 10, which is PCE2, a 512Kx8 memory mapped I/O space (only 4K used). We use 10:0000-10:0EFF, 10:0FXX-10:0FFF is reserved. I/O space is decoded into some 256 byte blocks such as FEXX, and some 128 byte blocks such as FC8X-FCFX. Some peripherals such as the DS1284 clock and calendar need a 256-byte I/O space while others such as the SBX connector require less.

The serial loader RX and WX commands will read and write this I/O space. As far as the controller is concerned there is no difference between *external data memory* and *external data I/O*. Hence the term “memory-mapped I/O”. On the other hand, serial loader *program space* commands such as erase and fill commands, and memory and address tests will not write into this reserved I/O space.

The uCAN2 controller provides a PCE2 chip select for all I/O space. In addition, PLD U10 provides chip selects at FAXX, and FBXX, plus RD and WR strobes at FDXX. **Header P23 provides access to all the memory mapped I/O signals which are available for your custom use.**

µCAN2 MEMORY MAPPED I/O TABLE

Note that only the lower 12 bits of the address are decoded, and qualified with PCE2. Therefore, memory mapped I/O space reappears every 1000H within all PCE2 pages. For example, CLK_CS_L can be addressed at 10:0EXX, 10:1EXX, 10:2EXX, and so forth up through 17:FEXX

Address Range (hexadecimal)	signal name at P23	Active level	Signal location	Bytes	Description
10:0000-17:FFFF	PCE2	LOW	P23-3	512K	PCE2: 512Kx8 memory mapped I/O space
<p>Note: For simplicity, and to save me a lot of redundant typing, or at least cutting and pasting, all addresses below show only the lower 12 bits. On µCAN2, only the lower 12 bits of the address are decoded, and qualified with PCE2. Therefore, memory mapped I/O space reappears every 1000H within all PCE2 pages. For example, CLK_CS_L can be addressed at 10:0EXX, 10:1EXX, 10:2EXX, and so forth up through 17:FEXX. So mentally prepend a "10:X - 17:X" to all the three-nibble addresses in the table to get the full 24 bit addresses.</p>					
F00-FFF	N/A	N/A	N/A	256	reserved by Systronix or decode it yourself
E00-EFF	CLK_CS_L	LOW	U10-15	256	DS1284 chip select at EXX used by UCAN2
D00-DFE and WR	DXX_WR_L	LOW	P23-8	256	write strobe at DXX available for your use
D00-DFE and RD	DXX_RD_L	LOW	P23-7	256	read strobe at FDXX available for your use
C80-CFF	MCS1_L	LOW	P23-6	128	SBX chip select 1 or available for your use
C00-C7F	MCS0_L	LOW		128	SBX chip select 0 or available for your use
B00-BFF	BXX_L	LOW	P23-5	256	chip select BXX available for your use
A00-AFF	AXX_L	LOW	P23-4	256	chip select AXX available for your use
external data READ	RD_L	LOW	P23-1		read strobe available for your use
external data WRITE	WR_L	LOW	P23-2		write strobe available for your use

■ *External Memory or Peripheral Devices*

??? edited to here

The demultiplexed low-order address lines A0-A7 are brought out to header P24. AD0-AD7 (MCU Port 0) are on header P20. The un-multiplexed high order address lines A8-A15 (MCU Port 2) are on header P21. These signals are all used to access memory on UCAN2. You can easily add external memory or peripheral devices to the prototype area, address them with A0-A15, and strobe them with WR, RD and PSEN.

Recommended Peripheral Addressing

Many peripheral devices use 8 address/data lines plus chip select and read and write strobes. If all eight address lines are used inside the peripheral, it will have a 256-byte address space. Therefore, it's convenient to only decode the upper byte of the HSM's address, into 256-byte I/O blocks at F0XX, F1XX, F2XX,... FFFX.

You will want to decode your device chip select into this uCAN2 address space at F000H - FFFFH to avoid conflicts with the UCAN2 NVRAM. We've provided the address decodes in the table above. If you wish, you can further decode all 16 address bits down to a single byte address. Be aware that adding more decoding slows down the timing of your chip select and will probably require stretch cycles when accessing that device. The decodes we've provided are done in high speed PLDs and do not require additional stretch cycles.

Create your own decoder PLD or ask about stock devices available from us.

Protecting Processor Pins from Static or Under-voltage

If you will be accessing machinery or equipment attached by cables to the address and data busses of the HSM controller, you will need to buffer them or at least protect them against static or under voltage conditions before they leave UCAN2. **Do not wire controller pins directly to a cable which drives a printer or other device.** Page 60 of the 1993 *Dallas Soft Microcontroller Data Book* shows a combination of schottky diodes and 1K ohm resistors to protect processor port pins from negative voltages. Other application notes on pin protection are available from Dallas Semiconductor and Systronix.

Alternatively, ICs are available with active clamping and current limiting. One such device is the Texas Instruments TL7726 hex clamp. As the name implies, it has six inputs which provide protection for six I/O lines. It is intended to be used with an external current limiting resistor on each input. Data sheets and application notes are available at the Texas Instruments web site, www.ti.com. Systronix stocks the TL7726 in both DIP8 and SOIC8 packages.

Testing After Adding Peripherals

The loader M and A commands provide memory and address line tests, respectively. It's a good idea to run these after you add peripherals to your system to verify that you haven't shorted address or data lines to power, ground, or each other. Of course if the shorts are really bad, the test won't run at all. So check your work as you go and run the test as you add each component.

■ *Voltage Monitor, Reset and NVRAM Control*

On board circuitry protects the NVRAM from invalid write cycles during power up and power failure conditions. Board reset occurs at Vcc-10% or 4.50 volts nominal. An early Power Fail Warning interrupt occurs when the unregulated power input drops below 6 volts. The NVRAM is forced into low-power data retention mode when Vcc drops by 5% and the controller is reset. It is necessary that the system software not write to SRAM after the PFW interrupt. The NVRAM data is maintained until its backup supply drops below 2 volts.

Maxcap

Program and data NVRAM is built using very high speed (20 nsec access time) SRAM in order to support 40 MHz controller operation. At the time of the board design this was the only memory fast enough. This design decision drove many other parameters. These high speed SRAMs are not available in a version with a "low-low" power standby mode of a few microamps. This rules out the use of a Lithium coin cell battery for memory backup - it would be rapidly depleted.

Therefore we used a Maxcap which is recharged whenever board power is available. The Maxcap should last indefinitely. Due to the high standby current of the SRAMs, their backup time is typically two minutes. This is long enough to survive typical AC power bumps or brownouts. *We assumed that the board would typically be used in a development environment where long program retention was not as important as the ability to operate at full speed.* This was a tough decision. We'd like to know what you think, since we will be making the same design decision on future development boards. What's most important to you in development boards?

Lithium Battery

The DS1284 clock and calendar is backed up with a lithium battery. The CR2032 lithium battery is sized to provide typically ten years or more of current. The lithium battery has a temperature range of 0 to 70 degrees C. Below 0 C the battery will freeze and clock and calendar contents will be lost. When the battery warms up it will operate normally. Although the battery will provide backup down to almost 2.0 volts, it is a good idea to replace the battery if its voltage at room temperature drops below 2.7 volts. Any good grade of battery is acceptable. Lithium CR2032 batteries are available from Systronix, at most camera stores, department stores and even many drugstores.

Test points TP3 and TP4 are provided to measure the voltage across a 100 ohm resistor in series with the lithium battery. With no external power source, this voltage is typically less than 50 nV, or a current of 500 pA to maintain the NVRAM. With power on, a current of typically 300 nA flows into the lithium battery from the board's power supply. This is normal and does not harm the battery.

Note that the battery must be installed in order to access the DS1284 clock and calendar. If the battery is not present, the DS1284 cannot be accessed, even if Vcc is available.

Voltage Monitors

The DS80C390 specifies 4.00/4.13/4.25 V minimum operating Vcc. The HSM (and DS5000 family) power fail warning occurs at 4.25/4.38/4.50 min/typ/max. Typical 5V components such as SRAMs are specified with 10% Vcc tolerance, or 4.50 to 5.50 volts operating voltage

In order for UCAN2 to function reliably, all of its components must be operating properly, not just the controller. Therefore, system operation must stop when Vcc is less than 4.5 volts. UCAN2 uses a "5%" supervisor chip whose reset threshold is 4.50 volts minimum. A 5% supervisor resets the entire UCAN2 system at 4.50/4.65/4.75 volts min/typ/max. A 10% Vcc reset supervisor would allow more time to operate as power is failing but marginally violates the specification of the SRAM and other chips. uCAN2 write-protects the NVRAM and resets the controller when Vcc drops to the reset threshold of the supervisor chip. To prevent a reset during a NVRAM write cycle, the controller must stop writing to NVRAM before Vcc decreases to the reset threshold. Note that read cycles while power is failing are actually OK and will not corrupt NVRAM contents.

PFW Interrupt and System Shutdown Time

???? PFW on C390?

To provide time for system shutdown, we would like an early power failure interrupt. Ideally we should monitor the voltage prior to the on-board regulator. The power failure warning interrupt built into the controller may never be triggered since its threshold (4.25/4.38/4.50 min/typ/max) is below a 5% Vcc reset threshold and identical to a 10% threshold. Therefore UCAN2's reset circuitry will be triggered before the 87C550's internal reset circuitry. The 87C550's built-in PFW interrupt has another limitation - it can only monitor Vcc since it is internally connected to the controller's power pin. The supervisor used in UCAN2 monitors the unregulated DC input and emits a PFW interrupt when the unregulated power input falls below about 6 volts (depending on load). This provides plenty of time for shutdown, since at this early state, regulated Vcc is still at its nominal 5 volt value.

PFW Interrupt Routine Tips

The exact amount of time available between the PFW interrupt and a controller reset depends on several factors: component tolerances, temperature, power supply capacitance (both pre- and post- regulator), and the total current consumption of your system. To be absolutely safe,

your PFW interrupt code should not perform any critical writes to the NVRAM. Finally, verify that your routine completes well before the worst-case Vcc reset threshold. You can use an oscilloscope to monitor the time at which your PFW routine drives an I/O device and the time at which the controller reset signal occurs.

■ *Interrupts and Timer/Counter Inputs*

The Dallas 87C550 has 16 interrupt sources with three priority levels. There are six external interrupts and ten internal ones. Of the external interrupts, interrupt 0 is the highest priority. When the external interrupt pins are not being used in their special modes, they can be used as general purpose I/O pins. For detailed interrupt information, and all internal interrupt details, please go to the DS80C390 data sheet.

External Interrupt Pins

Interrupts used by UCAN2 peripherals

INT0 is used by the UCAN2 power failure warning (PFW) signal, if JP25 is installed. INT1 is used by the DS1284 clock and calendar INTA or INTB outputs if JP1 is installed. INT2 and INT3 are routed through the normally connected jumpers JP15 and JP14, respectively. If there is no SBX card present or it does not use interrupts, INT2 and INT3 won't be driven by any part of the SBX hardware. INT4 and INT5 are not used by UCAN2 peripherals.

INT0/P3.2

(Active low, edge or level sensitive). External Power Failure Warning when the UCAN2 unregulated power input drops below approximately 6 volts. (Do not confuse this with the 87C550 internal PFW interrupt.) This should be the highest priority interrupt in your program. Jumper JP25 is provided to control the PFW-to-INT0 connection. You must install a jumper to enable the external PFW interrupt.

INT0 is also tied to jumper block P3. By default, it is not connected to the Low Level pushbutton switch S4. By placing jumpers on P3 you can connect INTO and/or INT1 to the pushbutton. The ability to drive more than one interrupt input simultaneously helps debug interrupt conflicts.

INT1/P3.3

(Active low, edge or level sensitive). Interrupt output from DS1284 clock and calendar. Jumper JP1 is provided to control which if any interrupt from the DS1284 is connected to the DS80C390. The silkscreen on the board identifies one end as INTA and the other as INTB. The center terminal of the 3-contact jumper goes to the DS80C390 INT1 input. If no jumper is present, no interrupt from the DS1284 is connected.

INT1 is also tied to jumper block P3. By default, it is not connected to the Low Level pushbutton switch S4. By placing jumpers on P3 you can connect INTO and/or INT1 to the pushbutton. The ability to drive more than one interrupt input simultaneously helps debug interrupt conflicts.

INT2/3/4/5

(Edge sensitive: rising, falling or both) **INT2 and INT3** are tied through JP15 (SBX Interrupt 0, DS80C390 INT2) and JP14 (SBX Interrupt 1, DS80C390 INT3). These jumpers are “normally closed” by a trace on the bottom side of the board. To disable this, cut the trace. You can then install a jumper block and a jumper if you later decide to re-enable them.

All **INT2/3/4/5** are tied to jumper block P2. By default, none of these inputs are connected to the High Level pushbutton switch S3. By placing jumpers on P2 you can connect one or more of the interrupt inputs to the pushbutton. The ability to drive more than one interrupt input simultaneously helps debug interrupt conflicts. If you aren't using the SBX interrupts (many SBX cards don't), there's no conflict with this pushbutton.

■ **Dallas MicroLan/iButton**

uCAN2 includes a simple interface to the Dallas One Wire network (DOW net), also called MicroLan. This is similar to the iButton protocol, with the difference that DOW devices are addressable and designed to allow multiple devices to share the same network wire. UCAN2 includes a Dallas DS1820 temperature sensor. An indexed PDF file data sheet is available from Systronix.

DS1820 Temperature Sensor

The DOW net connector is an RJ11 type. You can use standard telephone-type RJ connectors and cable, available from a variety of distributors.

Caution: The DOW_DQ pin is tied directly to the DS80C390 P3.4. Exposure to static discharges could permanently destroy that pin on the processor. You may want to add your own static protection to the DOW net.

We have included a simple DS1820 test program in HEX form (no source code). It supports reading the DS1820. It does not support reading multiple devices on a network.

A full-featured DOW network driver with source code and technical support will be available “soon” from Systronix as a commercial product.

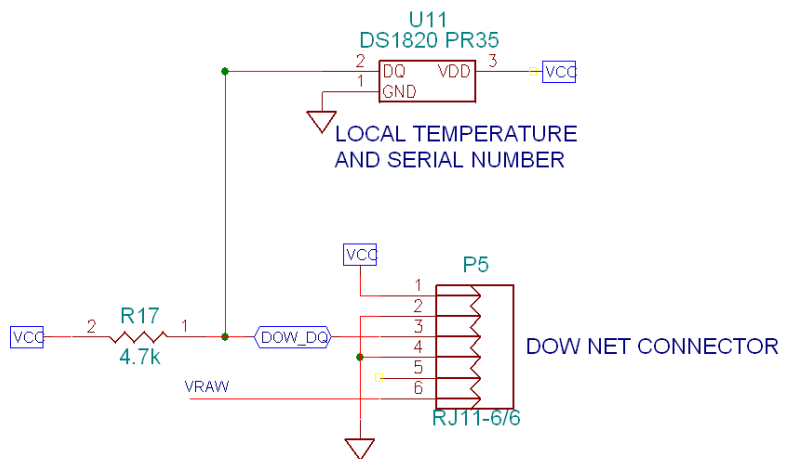


Figure 2. Header P5 and U11, DOW net

■ ***DS1284 Clock and Calendar***

The DS1284 is a clock and calendar chip which operates on its own crystal, independently of the DS80C390 controller. The DS1284 is backed up by the same lithium battery which preserves UCAN2 code and data memory. The DS1284 is memory mapped at address FEXX.

The DS1284 can generate periodic or calendar-based interrupts which can be jumpered to the DS80C390 interrupt input INT1. The DS1284 also can generate an accurate 1 KHz square wave, available on J2. The DS1284 includes some NVRAM which is independent of all other UCAN2 memory. This is a handy place to store constants or parameters which you do not want to be changed by frequent new program loads. The UCAN2 NVRAM is never cleared or filled by the loader.

We've included a sample program, 550_ECC.ASM, which reads and displays the time since reset.

The complete DS1284 data sheet is included as a PDF file, so we won't attempt to go into detail on using it here. We've added a detailed index so you can quickly locate specific areas of interest.

■ ***SBX Mezzanine Connector***

The SBX connector is a well-established, simple mezzanine bus connector. It's address space is limited, but the interface is simple and low cost. A variety of SBX boards are available from numerous vendors: if you are looking for some, start with a Web search.

Systronix SBX1 LCD and Keypad Board

This SBX card is an easy way to add the most common digital I/O to your system. SBX1 includes a 4x4 or 4x5 matrix keypad decoder/debouncer, a parallel interface LCD connector, LCD backlight control, a buzzer, 24 bits of bidirectional I/O, and a standard Opto-22 style 25x2 I/O header.

Systronix SBX-P Prototyping Board

Coming soon - this prototyping board is still in the concept stage. E-mail your suggestions to sbxp@systronix.com.

Using Other SBX Boards

You "should" be able to simply plug them onto UCAN2, write the necessary firmware and go. The I/O space read and write commands (RX and WX) in the UCAN2 loader are very helpful in debugging your SBX interface. Test out your idea manually or by sending a text file to the loader, then write the same algorithm into a program.

TP1 and TP2 are included to tie +/- 12V to the SBX connector if your card requires it. You will need to provide your own +/- 12V supply, there is none on UCAN2.

■ *Using and Modifying UCAN2 I/O and the Prototype Area*

UCAN2 is notable chiefly for its analog I/O. We've also provided some basic I/O such as dual RS232 ports, but you can easily modify these or add more to suit your needs.

LED Test Points

T2 and T3, if driven low, will cause LEDs D5 and D6, respectively, to light. Only 1 mA of sink capability is needed, so these test points can be driven by a controller port pin.

Adding other Serial I/O

There wasn't room on UCAN2 to provide the means to change the serial I/O easily. If you need additional serial I/O or another type such as RS485, we recommend adding a UART such as the Philips 2691 (available from Systronix) to the prototype area. Or plug on one of the available SBX cards with additional serial I/O. Or use an available converter (from companies such as Black Box at www.blackbox.com) on the existing UCAN2 serial output.

S3 and S4 Pushbuttons

All interrupts are disconnected from the pushbuttons unless the P2 or P3 jumpers have shorting blocks installed. If you aren't using the switches S3 and S4 for interrupts (or even if you are) you can connect them to devices in the prototype area. Refer to the interrupt section for details. Some interrupts may be driven by UCAN2 devices - if you do wire them also into the switches and other prototype circuitry, the UCAN2 devices will then also be driving your circuitry.

VCC and GND in Prototype Area

We've provided marked headers adjacent to the prototype area for easy VCC (5 volts) and ground access. In addition, the top board layer in the prototype area is VCC. All the exposed shiny diamonds on the top of the board in the prototype area are solder-plated VCC attachment points.

The bottom board layer in the prototype area is ground. All the exposed shiny diamonds on the bottom of the board in the prototype area are solder-plated ground attachment points.

DALLAS HIGH SPEED MICROCONTROLLERS

The Dallas High Speed Microcontrollers (HSMs) are supersets of the venerable 8051 microcontroller. Well over 100 million 8051s have been shipped, and they are in use in everything from DNA replicators to blood glucose meters. The HSM family is code-compatible with the 8051 instruction set. The HSM core architecture has been redesigned to offer faster execution while consuming less power.

■ ***High Speed Microcontroller Data Sheets***

A data book is available from Dallas Semiconductor. Call them at 972-371-4000, or better yet, get the data sheets today in PDF form at www.dalsemi.com. There is also a HSM data sheet link from our web site at www.systronix.com. Acrobat reader is free at www.adobe.com. Datasheets from Systronix require Acrobat 4.0 or later to display all the Acrobat features such as highlighting.

■ ***What's Different About the HSM Family***

Port 0

No Port 0 pullups are required or advised with the High Speed Microcontroller Family. Port0 has special drivers which can tolerate up to 100 pF and meet the required high speed timing when Port 0 is used as the multiplexed address/data bus AD[7..0]. In the case of address to ALE setup, this means charging or discharging this much capacitance in 11 nsec at 22.1184 Mhz. Adding Port 0 pullups significantly degrades signal integrity on Port 0, causing more overshoot and undershoot, and more noise during ALE transitions.

Memory Timing

There is more to memory timing than this manual has space to discuss. Overall memory system timing is a combination of memory device (both SRAM and EPROM), processor speed, bus loading, address latch, and decode circuit timing. These calculations can be quite complex and can trade off component speed versus cost and availability.

Memory timing application notes are available at the Dallas Semiconductor web site at www.dalsemi.com, and there are also links to these from www.systronix.com. Suffice it to say that at frequencies above 16 Mhz, a faster Port 0 demultiplexing latch is required.

UCAN2 uses an AC573. This device can produce large switching currents into capacitive loads. Low-value series termination resistors are advisable on its outputs to avoid excessive overshoot. For noise immunity, this latch should use CMOS input thresholds, never TTL.

To provide code access which complies with data sheet specifications, fast memory devices and fast control circuitry are also required. There is no provision to add wait states or stretch cycles to code access in the Dallas HSM family. UCAN2 uses a 10 nsec PLD to control memory access. All uCAN2 circuitry has been designed to provide reliable worst-case performance at full operating speeds.

Data access (both memory and I/O devices) can be slowed down with stretch cycles. All uCAN2 circuitry has been designed to provide reliable worst-case data access at full operating speeds with no stretch cycles added.

Strobes

Due to the fast transitions on strobes, termination may be required, depending on your design particulars. uCAN2 provides special termination of ALE for low emitted noise and reliable operation during Port 0 transitions.

Instruction Timing

The High Speed Microcontroller Family owes its performance improvements to a redesigned 8051 core. This core maintains code compatibility with the 8051 family while cutting the number of clock cycles per instruction cycle from 12 to 4. This gives a 3X increase in instruction cycles executed per second, assuming all instructions take the same number of instruction cycles. But they don't, so the performance difference is really between 2X and 3X for a typical instruction mix. Detailed instruction timing is available in the 80C320 family data book.

Power Supply and Reset Circuitry

Never, ever use a capacitor and resistor for reset of a HSM family processor. It's not even a good idea on a generic 8051. UCAN2 uses a precision reset control chip with proprietary Systronix circuitry to provide reliable reset control, NVRAM protection during power-up and power-down, and Power Failure Warning (PFW) interrupt generation.

Power supply accuracy and regulation with temperature and load variation is also critical. Low cost regulators with 2% or better tolerance are widely available, and are used in all Systronix products. We also use high-quality low-ESR electrolytic capacitors for bulk power supply bypassing. Bypass capacitors at the processor, memory, and high-speed support chips must be high-quality tantalum and multi-layer ceramic devices.

Additional Features

The High Speed Microcontroller Family has additional peripherals or memory, depending on the family member. Most members include a second UART, watchdog timer, and additional external interrupts. Some include 1 KByte of on-board movx memory.

TROUBLESHOOTING & DEVELOPMENT TIPS

No Serial Communication between PC and uCAN2

This is the most common problem. The serial port on UCAN2 is very simple and robust - it's very hard to damage it. If your PC isn't talking to UCAN2, it's most likely that the problem is in your PC or the cable between your PC and UCAN2. The serial port on UCAN2 does not use hardware flow control, so presence or absence of handshaking signals from your PC has no effect on UCAN2.

1. Often, people connect uCAN2 to a previously unused serial port on their PC.
 - a. If possible, use a PC COM port which you know has recently been operating correctly with another known good serial device such as a modem or printer.
 - b. If you have a serial mouse or pointing device and you know it works, swap your mouse COM port to the unused port, and put UCAN2 on the former mouse port. You will have to change your PC's configuration in order to do this.
2. On a DOS or Windows 3.X PC, another I/O driver may be loading and interfering with the PC serial port you're trying to use.
 - a. Check your PC's setup
 - b. Try another PC
 - c. Try using another, known good COM port on your PC.
3. You must use a straight-through cable (not a null modem cable).
 - a. This means pin 2 is your PC's RXD (UCAN2 TXD) and pin 3 is the PC's TXD (UCAN2 RXD). Ground is pin 5. The pins are usually numbered, molded into the plastic inside the DB9 shell (it's very tiny print!). The schematics contain a detailed pinout of the serial connector.
 - b. A null modem cable has TXD and RXD swapped within the cable to permit connecting 2 PCs together, as if there were a modem between them (hence the name "null modem"). You can't use a null modem cable with UCAN2.
4. If you have an oscilloscope or logic probe, connect it to RXD1, Port1.6 This is labeled on the header P4-6. This signal is the serial input to the controller, it is a CMOS level, and is not the RS232 voltage level. At 19.2 kbaud, bits are 52 usec wide, so set the oscilloscope time base to 50 to 100 usec per division. Now in load mode, when you press your PC's Enter key, you should see periodic characters on RXD1. This is your PC sending carriage returns to UCAN2. HSM should respond with a burst of characters on Port1.7, header P4-7, the controller TXD1 output. These characters are the loader printing the opening prompt to UCAN2 COM1. This verifies that the loader is receiving and sending serial information from/to the RS232 level translator.

- a. You must push the load pushbutton and hold it for more than 500 msec to trigger load mode. Each time you push the button in this manner, and send the loader a carriage return character (ODH), the loader should emit a brief burst of serial characters, the opening prompt. If you hold down the enter key the loader will continue to emit loader prompts. If TXD1 stays high, then for some reason the loader is not starting up.
 - i. Check the board power at a VCC and GND test point. Is it getting 5 volts +/- 5% (4.75 to 5.25 volts DC)?
 - ii. Is the ALE strobe oscillating? This indicates the processor is operating. If not, is the crystal seated in its socket? Absence of ALE indicates an inoperative controller chip, most likely due to a power or crystal problem.
5. Next check the RS232 level translator: carefully probe pin 14 of U16, the SP312A serial input from your PC. Pin 15 of U16 is the serial output from UCAN2. BE CAREFUL! If you short the charge pump pins to data pins with a scope probe, you could permanently damage the SP312A device and/or the UCAN2 controller. This is because RS232 voltage levels exceed safe CMOS or TTL levels.
 - a. If serial input and output on the PC side of the SP312A are toggling between at least +5V and -5V with the cable connected, then RS232 serial data is getting out of the UCAN2 board and the problem is in your PC's serial cable or serial port, or the configuration of your PC terminal software.
 - b. If RS232 output from the PC is toggling and the TTL signal to the controller is not, then the SP312A level translator is damaged. Likewise, if serial data is toggling from the controller but not appearing on the output of the level translator, it may be damaged or your PC or cable may be clashing with the RS232 signal. Remember that you will not see any output from the loader unless it receives a carriage return.

In any case, PLEASE CONTACT US before you return your board. We have almost a 100% success rate in solving problems like this over the phone. We do charge a minimum service fee on boards which are returned without authorization and check out to be operating fine. So please save yourself and us time and aggravation and call us first. We'll do our best to get you up and running right away. If you need to return your board, you need an RMA (return merchandise authorization). If you have a major credit card we can ship a replacement board at once without first receiving your old board back for evaluation.

Internet FAQ

There is a FAQ (Frequently Asked Questions) on our web site. It is updated on a regular basis.

Start Simple

Start with a simple program, get it working, and then add complexity in modules. Try to add new functions as subroutines which you can call or not call to easily isolate suspected problems.

Learning Assembly Code and Embedded Programming

If you find a simple way to learn assembly code or embedded programming, let us know! We get asked “how do I learn ...” quite frequently. Maybe in a few years there will really be such a science as Computer Science, but at the moment, starting a discussion about “good design” is a good way to start an argument. This manual is not intended to teach you assembly code or good programming principles. Here are a few good books. Some of these may be hard to find - good luck - we don't stock them for that reason. If your local big chain bookstore won't order them, try a college bookstore or a smaller, independently owned bookstore, or search for the title on the Internet.

The Art of Embedded System Programming by Jack Ganssle, published by Academic Press. This one should be easy to order at your local bookstore. This book focuses on philosophy and general good programming hygiene. It's not specific to any processor. Lots of true and humorous anecdotes, very readable. He also publishes a delightful free e-mail newsletter. To subscribe, send a message to majordomo@ganssle.com, with the words "subscribe embedded your-email-address" in the body. Of course, substitute your actual email address for the text 'your-email-address'.

The Microcontroller Idea Book by Jan Axelson. YES we do stock this one. It's a good overview of microcontroller interfacing ideas, intended for hobbyists or those who are new to embedded systems. Lots of schematics and sample code (available on disk) for BASIC-52 controllers.

C and the 8051 by Thomas Schultz. As the title suggests, Professor Schultz assumes you will be using C, so most of the source code is in C. But there is a lot of good information applicable to any language or assembly code. Available at major bookstores and www.amazon.com.

Exception Handling

All embedded applications should have run-time *exception handling*. Good exception handling is one hallmark of a good programmer. Exception handling is a difficult topic, and highly application dependent, so it is hard to make specific recommendations. The serial loader uses exception handling to deal with bad command lines, bad hex files, and similar situations.

If you are using BCI51 Pro, please, please take advantage of the BCI51 ONERR instruction to provide run-time error handling. Print the error code to the serial port so that you at least know what error occurred.

Quick Diagnosis Table

UCAN2 QUICK DIAGNOSIS TABLE		
SYMPTOM	EXPLANATION	SOLUTION
RUN and LOAD LEDs are both off	no power to UCAN2	check power cube - center is positive VCC-GND short in prototype area could be causing regulator shutdown.
No prompt from loader on PC	no serial communications from UCAN2 to your PC	press and hold LOAD button, RED LOAD LED should be lit to confirm LOAD mode. serial cable must be straight-through is PC serial comm software properly configured? You must send a carriage return (ODH) to the loader to establish communications. Start with 19200 baud.
I added parts to prototype area, now operation of the loader is erratic	Overloaded or shorted address or data lines	If loader won't even run, beep out address and data lines - look for shorts to each other or to power or ground in the circuitry you've added.
I put in a faster crystal, now the board doesn't work reliably	33 MHz is the fastest crystal.	Don't do this! UCAN2 will not run faster than 33 MHz.
I put in a slower crystal, now the board doesn't work reliably	Crystal may be wrong cut or load capacitance for UCAN2.	The 8051 requires a parallel resonant, fundamental mode crystal with about 18 pF load capacitance. Not all crystals can generate all baud rates.
Loader starts up, displays help, but I get HEX file load errors	Loader EPROM address and data lines are OK, but not NVRAM address and data lines	Run loader M and A commands to test memory and address lines. If M command fails, run A command and beep out the address or data lines of circuitry you've added.
A simple "hello world" 8051 program I have doesn't run on UCAN2.	DS80C390 registers and interrupt vectors differ from those in a generic 8051 and even other High Speed Micros such as the DS80C320.	Modify the program to use the SFR and interrupt vectors of the DS80C390. Check the bookmarks in the PDF data sheet to quickly find areas of interest.
Serial I/O is garbled	Baud rate mismatch or PC serial comm program got "out of sync"	HyperTerminal in particular seems prone to lose sync with incoming data if it ever gets a partial character or data at the wrong speed. Close HyperTerminal and restart it. Get a free new version of Hyperterminal PE over the Internet.
I didn't use UCAN2 for a long time and my program is no longer in memory	Lithium battery ran down and you lost your program in NVRAM.	The Lithium battery should be good for five years at room temperature. High moisture or conductive atmosphere such as salt spray will increase battery leakage. Replace with a CR2032 3V battery when the voltage is less than 2.5 volts. Don't overload the Lithium battery output in the prototype area.

UCAN2 QUICK DIAGNOSIS TABLE		
SYMPTOM	EXPLANATION	SOLUTION
BCI51 program runs then appears to hang	Run time error may have occurred (divide by zero, math overflow, etc)	Use ONERR to print the error value to the serial port, or blink an LED with a pattern so that you know when a runtime error occurred.

■ **Warranty**

uCAN2 is warranted against defects in manufacturing for a period of one year. This does not include damage due to static, driving controller I/O pins beyond their design limits (see the DS80C390 data sheet for pin specifications) or other damage caused by improper use. Systronix is not liable for latent bugs, if any, in micro-controllers. In the case of a defective controller or processor (such as the infamous Intel floating point bug) we will honor all manufacturer's warranties and make every reasonable effort to keep your system up and running.

PLEASE CONTACT US before you return your board. We do charge a minimum service fee on boards which are returned without authorization and check out to be operating fine. So please save us both some possible aggravation and contact us first. We'll do our best to get you up and running right away. If you need to return your board, you need an RMA (return merchandise authorization). If you have a major credit card we can ship a replacement board at once without first receiving your old board back for evaluation.

uCAN2 SCHEMATICS

Replace this page with the uCAN2 schematics in the printed version of the manual.

Schematics are not available in the Web version of the documentation.

Schematic page 2/3 - controller and memory

Schematic page 3/3 - serial I/O