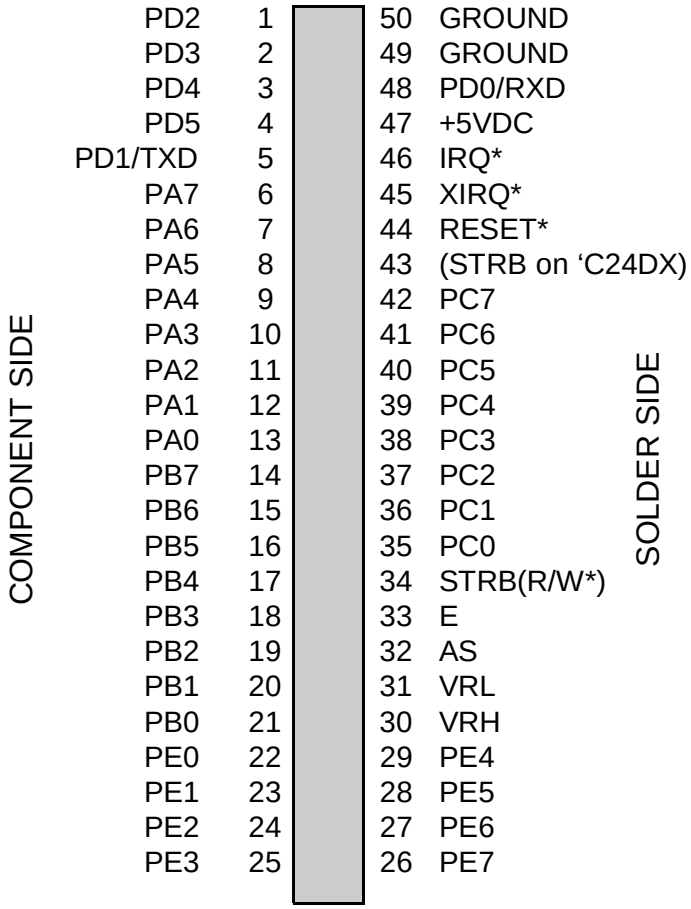
APPENDIX A
Pinouts for SB Connector Option
Adapt11 and Adapt11C24DX



AERIAL VIEW

Note: * indicates 'active low'

1 INTRODUCTION

1.1 Congratulations!

You are now the proud owner of **Adapt11**, a unique and useful tool for working with 68HC11 microcontrollers! Your questions and comments are always welcome. We provide friendly, knowledgeable technical support by telephone, fax, and e-mail to all our customers. As well, we have a comprehensive website with a resource page featuring new information, software, and links to other useful sites on the Internet. See back cover for how to contact Technological Arts.

1.2 Purpose of Adapt11...

Adapt11 was designed as an evaluation and application tool for various members of the Motorola MC68HC11 microcontroller family. It is unique among evaluation boards in that it is designed to plug vertically into any standard solderless breadboard. It is a fully functional, standalone implementation of a 68HC11 configuration, and can be treated just like a chip in your breadboard. Then it is simply a matter of wiring up the desired application circuits and downloading the appropriate code into the micro to evaluate and develop your application ideas.

1.3 Product Description

Adapt11 Starter Packages come with on-chip or on-board EEPROM program memory (depending on version) that is directly loadable via your PC serial port for quick and easy programming. The 68HC11's handy BOOT mode makes separate programming hardware unnecessary. All

Adapt11 modules will work with any A-series and E-series 68HC11 chips that are packaged in a 52-pin PLCC (see Table 1 for some examples). In addition, they all provide EPROM programming support the 68HC711E9 and 68HC711E20, in both UV-erasable and one-time programmable (OTP) versions. The user need only supply the proper Vpp programming voltage via the External Power connector. See section 4.3 for complete details

1.4 Description of Product Configurations

The original **Adapt11** Starter Package includes a 68HC811E2, with 2K EEPROM on-chip. Operating in single-chip mode, all I/O ports are available for user applications. When operated in expanded-chip mode, additional memory can be added externally; however, two of the I/O ports form the address and data buses, and are thus unavailable for user applications.

A couple of memory expansion cards for **Adapt11** are available. The first (MX1) is offered in two configurations: AD11MX1 provides 32K RAM (lower half of memory map) and an empty 28-pin DIP socket mapped to the upper half. You can plug an EPROM, EEPROM, or RAM chip in this socket for even more memory. The board is also offerd with a 32K EEPROM device in the socket (Adapt11MX1E). Both configurations include address demultiplexing circuitry, and can be coupled to **Adapt11** using a mini backplane/adapter, a stackable connector arrangement, or a standard 50-pin ribbon cable (eg. SCSI cable).

A second memory board (MX2) is identical to the first, except it adds battery-backup for the RAM and a clock/calendar circuit that is included on the board. This board will especially be of interest to those who wish to implement a datalogging application.

MCU	RAM	EEPROM	EPROM
68HC11E0	512	0	0
68HC11E1	512	512	0
68HC811E2	256	2K	0
68HC711E9	512	512	12K
68HC711E20	768	512	20K

TABLE 1 - MCU MEMORY RESOURCES

For those applications requiring more memory resources while keeping all the I/O ports, there is **Adapt11C24DX**. It combines Motorola's 68HC24 PRU with a 68HC11 and 8K or 32K EEPROM, offering full register compatibility with the 68HC11. From both a hardware and software point of view, this board can be thought of as a single-chip 68HC11 with 8 or 32K EEPROM and 512 bytes RAM. This combination is particularly well-suited to users who are developing applications for eventual use in a single-chip OTP-based design (eg. 68HC711E9). Software developed using **Adapt11C24DX** can be directly "burned" into the OTP PROM of the '711E9 destined for the target single-chip application. For '711E9 applications that will be using on-chip EEPROM, a 68HC11E1 can be used to pro-

Starter Package Type	Internal RAM (Start address)	External RAM Start address)	Internal EEPROM (Start address)	External EEPROM (Start address)	Replacement Port Type (Register Block)	MCU Type supplied in Starter Package
Adapt11	256 ($0000)	0	2K ($F800)	0	none	68HC811E2
Adapt11EVB	512 ($0000)	0	512 ($B600)	none (BUFFALO in ROM)	none	68HC11E9B
Adapt11C24DX8K	512 ($0000)	0	0	8K ($E000)	68HC24 ($10xx)	68HC11E0
Adapt11C24DX32K	512 ($0000)	0	0	32K ($8000)	68HC24 ($10xx)	68HC11E0
Adapt11C24DX60K	512 ($0000)	32K ($0000) with 4K "hole" at $2000	0	32K ($8000)	68HC24 ($10xx)	68HC11E0
Adapt11C24DXEVB	512 ($0000)	32K ($8000)	512 ($b600)	none (BUFFALO in ROM)	68HC24 ($10xx)	68HC11E9B

TABLE 2 - Starter Package Memory Configurations & Address Maps

vide the 512-byte EEPROM block, resulting in virtually true emulation.

The board is also offered in a 60K version, which has a 32K RAM chip "piggy-backed" on top of the 32K EEPROM. A 4K block of the RAM is "decoded" out of the memory map, to allow for the 68HC24 registers, so the available RAM is 28K. This configuration is useful when the on-chip 512 bytes of RAM is not adequate for the application.

Table 2 lists the standard Starter Package configurations available at the time this manual was printed. New products are being added all the time, so be sure to check the website regularly. If you subscribe to the techart-micros internet-based discussion group, you will automatically receive notification of new products and options as soon as they are released.

1.5 Communications

An RS-232-compatible 3-wire serial interface port (RX, TX, and Ground) is built into all Adapt11 variants, allowing communication with a PC, or any other device which has an RS-232 serial port. The EVB versions of the board have a 9-pin D-sub connector, The logic-level RXD and TXD signals from the MCU are also brought out to the 50-pin header, for applications such as RS-485 or MIDI.

1.6 Adapt11 Versus Other Evaluation Boards

There are many evaluation and development systems available. Most of them try to be universal in their application, containing every imaginable kind of support circuitry on-board, or providing a large area for prototyping by soldering or wire-wrapping parts on the board. The designer

Design with Microcontrollers
- John B. Peatman (professor at Georgia Tech)
- ISBN 0-07-049238-7
- This book is on a more advanced level. Uses both the 68hc11 and Intel 8096 as example systems.

Embedded Systems Programming in C and Assembler
- John Forrest Brown
- ISBN 0-442-01817-7
- Van Nostrand Reinhold, 1994 - 304 pages, $49.95
- covers Motorola and Intel processors

Microcomputer Engineering
- Gene H. Miller
- ISBN 0-13-584475-4
- 1993, Prentice Hall, Englewood Cliffs, NJ 07632
- Explains basics. Many clear, concise examples.

Microcontroller Technology, The 68HC11
- Peter Spasov
- ISBN 0-13-583568-2 - Prentice Hall

Microcontrollers: Architecture, Implementation, & Programming
- Kenneth Hintz and Daniel Tabak - ISBN 0-07-028977-8
- 1992, McGraw-Hill Inc.

Mobile Robots: Inspiration to Implementation
- Joseph L. Jones and Anita M. Flynn - Very hands-on book.
- Focuses on every detail involved in design & construction of "Rug Warrior", based on 68HC11A1, using Interactive C compiler.

Programming Microcontrollers in C
- Ted Van Sickle - ISBN 1-878707-14-0
- 1994, HighText Publications - 394 pages, $29.95
- thorough tutorial on C programming, covers aspects of C programming specific to embedded systems

MC68HC11: An Introduction
- Han-Way Huang - West Publishing
- ISBN 0-314-06735-3

Single- and Multiple-Chip Microcomputer Interfacing
- G. J. Lipovski - 1988, Prentice-Hall
- ISBN 0-13-810557-X 025
- 68HC11-specific textbook with examples & problems

5.0 SOURCES

Internet Resources

•Technological Arts: Check here often for new product info, new utilities, tips, applications information, and resources on the web.
website: www.technologicalarts.com
e-mail address: support@technologicalarts.com
•Motorola Freeware: http://freeware.aus.sps.mot.com/freeweb/
•68HC11 FTP Site: ftp://ftp.stack.urc.tue.nl/pub2/scrumpel/
•University of Alberta: ftp://ftp.ee.ualberta.ca

Publications

Motorola Fax-on-Demand: (602) 244-6609 or 800-774-1848
Motorola Semiconductor Literature Distribution Center
P.O. Box 20912, Phoenix, AZ 85036 1-800-441-2447

• Motorola Microcontroller Development Tools Directory
(MCUDEVTLDIR/D)
• M68HC11 Reference Manual (M68HC11RM/AD)
• M68HC11 E-series Technical Data book (MC68HC11E/D)
• MCU Toolbox (MCUTLBX/D)
• Single- and Multiple-Chip Microcomputer Interfacing
(Prentice-Hall or Motorola TB316; Lib. of Cong. Cat. # 87-60656)
• Motorola Freeware PC-Compatible 8-Bit Cross-Assemblers User's Manual (M68FCASS/AD1)
• PCBUG11 User's Manual (M68PCBUG11/D2)

Other Books

The 68HC11 Microcontroller
- Joseph D. Greenfield (at R.I.T.) - ISBN 0-03-051588-2
- 1992, Saunders College Publishing, (Harcourt Brace Jovanovich)

Data Acquisition & Process Control w/ the M68HC11 Microcontroller
- Frederick Driscoll, Robert Coughlin, Robert Villanucci of Wentworth Institute of Technology.
- 1994, Macmillan Publishing Company
- ISBN 0-02-33055-X
- example applications of interfaces to various sensors.

tried to second-guess what you might want to do, and as a result you pay more and have less flexibility. We chose instead, to take a modular approach. With **Adapt11**, all I/O lines and control signals are brought out to a standard 50-pin interface connector. With several different connector options available, you can use the module in whatever way best suits your needs. With the solderless breadboard header, you can treat the module like a big chip, and plug it right into your breadboard. Forget soldering or wire-wrapping– get started developing your application right away. Your prototyping space is virtually unlimited, using solderless breadboards! When you've got a design working and you're ready to move to something more permanent, Adapt11 Modular Prototyping System (AMPS) accessories give you the ability to easily build fully customized, compact applications at low cost, with a full range of accesories including backplanes, prototyping cards, memory expansion, and application-specific cards.

2 USING ADAPT11 WITH SOLDERLESS BREADBOARDS

Note: In descriptions throughout this manual, "Adapt11" refers to any member of the Adapt11 family, unless specified otherwise.

The standard connector option installed on Starter Package boards is the "SB" style, designed to plug into a solderless breadboard. If your board has a different connector option, you may still be able to use it with a solderless breadboard, by way of a 50-pin solderless breadboard adapter. See the Accessories page on our website, or contact us, for further information on these adapters. For contact information, refer to the back cover of this manual.

CAUTION!
Never insert your module into or remove it from a "live" breadboard. Make sure the power is OFF !

1) Any breadboard will do; however, you will find that the kind made with a softer, more pliable plastic (such as nylon) will be easier to use and more durable. Avoid excessive removal and insertion of your board, to extend the life of your breadboard.

2) When plugging **Adapt11** into your breadboard, keep it vertical and press gently but firmly, rocking the module back and forth slightly, until the pins are seated in the sockets. Use the same side-to-side gentle rocking motion, while pulling gently upward, to
remove the board.

3) Plug **Adapt11** into the middle area of your breadboard strip to allow maximum access on each end to all the signals. If possible, place an additional breadboard section in parallel on each side of Adapt11 for easier wiring of your circuits. *(HELPFUL HINT: If you are using the Analog inputs, make sure to wire your analog circuits as close to these pins as possible, to keep noise
levels down.)*

4) Choose a convention for wiring your power distribution buses. A logical approach is to make the inside bus logic 5V, and the outside buses GROUND. Never supply external power via J1 if you are supplying 5VDC via the breadboard connector pins. However, always connect the breadboard GROUND to Adapt11 GROUND.

5) ***CAUTION: Always keep the VPP/NRML switch at***

BUFFALO commands, type Help. Refer to Motorola's documentation on BUFFALO for more details. To implement the Trace command, you'll need to connect the MCU's XIRQ pin to OC5 via I/O connector H1 (on your breadboard). If PE0 is pulled high during reset, BUFFALO will jump to a user program in EEPROM.

4.7 Troubleshooting

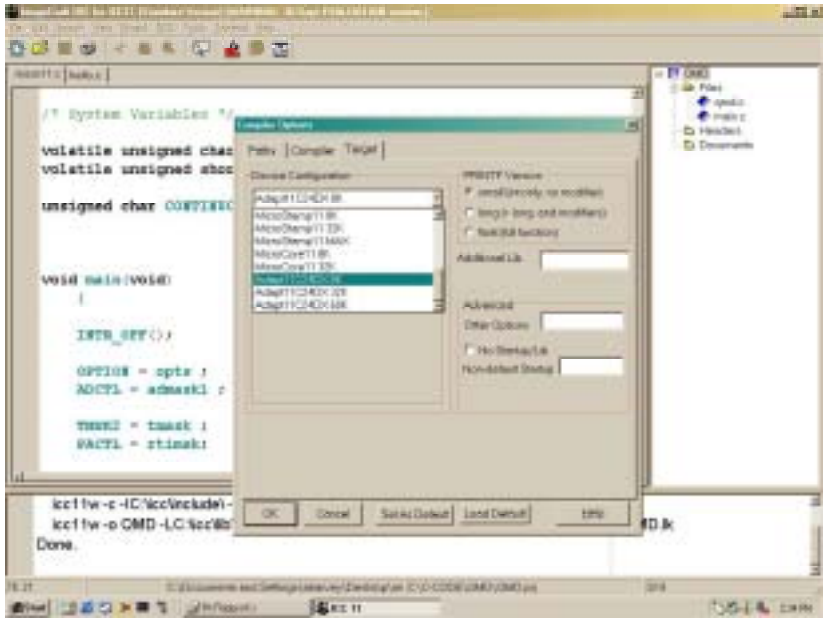
If you experience any problems using your Adapt11 product, be sure to check the Tech Support page of our website for assistance. A number of documents and resources are available there to help you. If you think you may have faulty *hardware*, check the relevant documents and follow the recommended steps. If you still can't resolve it, send a detailed description of the problem to support@technologicalarts.com. Make sure to mention whic product you are using. For all other issues (eg. application hardware and software design problems, third-party utilities, assemblers, compilers, tools, etc.) we recommend you join one or more of the internet-based discussion groups mentioned on the Tech Support webpage.

Our own group, **techart-micros**, is a forum for discussing various software tools available with boards made by us, any applications you may be working on or interested in, getting help with debugging your software, and to receive news from us about bugs, upgrades, and new products.

Motorola's 68HC11 discussion list has probably the world's largest group of 68HC11 users, and ranges from total beginners to advanced users and Motorola engineers. It is highly recommended for 68HC11-specific questions.

ImageCraft's discussion group is for all customers using their ICC11 C compilers. Look inside your C compiler's manual for instructions on how to join this group.

only time you would *not* need **vectors.c** is if you had a resident monitor that was controlling the loading and execution of a user program (such as the Buffalo monitor from Motorola).



Version 6 users will find that the user interface is somewhat different than described and pictured here, but the essential information of what settings need to be is the same. Refer to your ICC11 manual for more information.

4.6 Using BUFFALO

If you are using an EVB version of Adapt11, Motorola's BUFFALO debug/monitor program will be running from ROM. Connect the serial cable to your PC's COM port and use a terminal program set for 9600 baud, no parity, 8 data bits, 1 stop bit, and no handshaking. Make sure the switches on the board are set for RUN mode, NRML, and WRITE so that you'll be able to load and modify programs into RAM. To see a list of the

NRML unless you are programming an EPROM-type chip (68HC711E9 or 68HC711E20). See section 4.3 for more information on the purpose of this switch.

6) If you are using voltages other than 5V, make sure to keep these well away from Adapt11 pins and tie-strips, to avoid accidental shorts which may damage the module.

3 TUTORIAL

Note that this manual is not meant to provide an exhaustive study of the 68HC11 family, but rather to help you get started using the Adapt11 series of microcontroller boards as a learning and application development tool for 68HC11, whether you're a beginner or an expert. If you are a beginner, you will benefit from additional material listed in the Reference section of this manual, and links provided on the Resource page of our website (see back cover for URL).

CAUTION!
Never insert Adapt11 into or remove Adapt11 from a "live" breadboard. Make sure the power is OFF !

3.1 Getting Started

Important! Be sure to browse the README.TXT file on your Starter Package disk for information on what's on the disk and how to use it. Use any text editor or browser you like, such as Notepad or DOS edit, and follow the instructions for copying the disk contents onto your hard drive.

All versions of Adapt11 (except EVBs) have a demonstration program already programmed into the EEPROM when you receive it. This is a useful program for testing your communications setup and monitoring & controlling the various I/O lines of the micro.

You can power the module in one of two ways:

1) supply power via the external power connector; just connect a DC voltage greater than 5.6 Volts to the external power connector on Adapt11. Red is positive, and black is negative (ground). ***CAUTION! Make sure you have the polarity correct!***

2) supply regulated 5VDC via the appropriate pins on the 50-pin connector (H1). See Appendix A for the pinout diagram of Adapt11. ***CAUTION! Double-check your connections before applying power!***

To use the demo program, connect the supplied se

```
ADAPT11 DEMO PROGRAM COMMAND MENU

A  => SHOW PORT A STATUS
B  => CLEAR PORT B OUTPUTS
C  => SHOW PORT C STATUS
D  => SHOW PORT D STATUS
R  => SHOW REAL-TIME ANALOG VALUES
S  => BEEP SPEAKER (CONNECT TO PA6)
0 TO 7 => TOGGLE THE SELECTED B PORT LINE
 ?
<A>
 PORTA=000
<B>
 PORT B CLEARED
<C>
 PORTC=000
<D>
 PORTD=001
<R>
```

Figure 3.1 Adapt11 Demo Software Menu as it appears in a terminal program's window.

Download, and select the s-record file you wish to download. Press your board's Reset button and, where applicable, slide the Write Protect switch to WRITE. Then click OK.DOS environment. If you have purchased the Windows version, however, the following section provides some basic hints and guidelines for setting it up and using it with the Adapt11 family of products.

The LINKER is the part of a compiler package that marries the software with the particular hardware configuration being used. Therefore, you need to setup the LINKER before using ICC11. Find the Linker setup tab by pulling down the OPTIONS menu to COMPILER. In the Linker, "text" refers to code, and should be the starting address of EEPROM (or wherever you plan to put the code); "data" is for variables, and should be the start of RAM; "stack" should be the end of RAM, since it builds "downward". There are some exceptions: if you have a large amount of RAM, you would probably start the "data" section following the last address of on-chip registers, and put the stack at the top of internal RAM.

If you have ICC11 V4.5 or higher, you can use the Set upWizard feature of the Options|Compiler|Linker pulldown menu. For Adapt11, select "Generic 6811E2 single-chip mode". If you have V6 or higher, just select the product name from the list.

Unless you are using an EVB (with BUFFALO monitor in ROM), the MCU won't know after reset where your program starts. It looks in a special location in memory (0xfffe) called the Reset Vector to get the address of the start of your program. To make sure your program's starting address gets put there automatically, every C program you write will need to have the file **vectors.c** included (either explicitly at the end of the program, or as one of the files in the Project if you are using the ProjectBuilder feature of ICC11). You should examine and compile the program **hello.c**, in the Examples directory of ICC11. This program has the essentials set up for you. The

3) Any Adapt11 series board using 68HC711E9 (12K EPROM):

sbasic infile /cd000 /v0000 >outfile.asc

4) Any Adapt11 series board with 32K EPROM or RAM in the upper half of the memory map:

sbasic infile /c8000 /v0000 >outfile.asc

The resulting file **outfile.asc** is in assembler source code, and can be viewed and edited if you wish. When ready, assemble the file to produce an s-record file suitable for downloading to your board. Use the assembler provided with SBASIC, as follows:

asmhc11 outfile.asc

and it will produce the file **outfile.s19**.

4.5 Using ImageCraft's ICC11 Windows C Compiler

ImageCraft offers a good low-cost ANSI C compiler, which is quite popular in the 68HC11 community. A Freeware DOS version is included on your Starter Package disk, and is intended for those who are quite familiar working in C within a

To compile and download your file (for example, **hello.c**), select "Compile to Executable" in the pulldown menu. ICC11 will compile, assemble, and link, creating an s-record file called **hello.s19**. This is the file you will download to your board.

Make sure Bootstrap Download Mode is checked in the TERMINAL pulldown menu. Then activate the terminal window, and click on Bootstrap Options.

Select the appropriate Bootloader Programming setting for your hardware configuration: Internal EEPROM for Adapt11, External EEPROM for all others Adapt11 configurations.

Leave the Config Reg. setting at 0 (ie. "don't change") unless you want the bootloader to change the value of the Config. register. (Remember, the new setting only takes effect following a Reset of the HC11).

After exiting Bootstrap Options, click on Bootstrap

rial cable between Adapt11 and a serial port on your PC. (With some PCs, you will need a 9-pin to 25-pin adapter.) Run a terminal program (such as ProCommPlus, or the Windows Terminal program) on your PC, in your terminal program, set the baud rate to 2400, parity to NONE, # DATA BITS = 8, and #STOP BITS = 1. With SW2 set to RUN, press the Adapt11 RESET button. LED D1 will blink twice, indicating the demo programming is running. Hit <ENTER> on your keyboard. A menu of commands will appear on your terminal window's screen, followed by a command prompt "?" symbol. Each command is activated by a single keystroke. A sample screen showing results of each command is shown in Figure 3.1. Typing a command not listed will cause the menu to be re-displayed.

The A, C, and D commands in the demo program allow you to examine the states of PORTA, PORTC, and PORTD. Try putting switches on some of these input port lines. Use a 10K or higher pullup resistor on one side and connect the other side of the switch to ground. Note that PA0-PA2 are inputs, PA3 and PA7 are programmable as input or output, and default to inputs. PA4-PA6 are outputs only. In the demo program, PA6 is used as a tone output for a speaker. It is also connected to LED D1, to provide a visual output. You can drive a small piezo speaker directly by hooking one end to PA6 through a 330-Ohm resistor, and the other end to ground. When you press RESET, or type "S" when the demo program is running, you will hear two beeps from the speaker (or the LED will flash twice).

PORTB is an output-only port on the Adapt11. On Adapt11C75 and Adapt11C75DX, it is renamed XPORTA, and is bidirectional. In the demo program, however, it is set up as an output. Typing a digit between 0 and 7 causes the output state of the corresponding PORTB (or XPORTA) line to be toggled (eg. typing 3 causes PB3 (or XPA3) to flip to a

high if it was low previously, or a low if it was high previously). This allows you to activate LEDs (when driving LEDs directly from an output port, limit the current to a maximum of 10mA with 330-Ohm current limiting resistors on each LED); or drive relays, solenoids, or motors (with appropriate driver circuits). Typing B forces all PORTB (or XPORTA) output lines low. Typing R causes the values of all 8 analog-to-digital converter (ANALOG) channels of PORTE to be continuously updated on the screen (near Real-time updates) The display will continue to be updated until a key is pressed. Analog channels (ANALOG0-ANALOG7) can read voltages between 0 and 5 Volts. Try putting a 10K-Ohm (or higher) pot across the VRL and VRH pins (pins 30 and 31), and connect the wiper to an ANALOG input through a 1K-Ohm current limiting resistor; then change the pot setting, monitoring the ANALOG values on the screen. Unused ANALOG channels should be grounded to VRL through a minimum 1K-Ohm resistor. These inputs are not internally protected from electrostatic discharge (ESD) as the other input port lines are. (HELPFUL HINT: Grounding multiple adjacent analog inputs is easy by plugging a bussed resistor SIP in your breadboard and jumpering the SIP common pin to VRL.)

3.2 Writing Your First Program

If you are already experienced with the 68HC11 family of microcontrollers, you can skip this section. However, you may find the suggestions here useful to familiarize yourself with the essentials of getting a program to work the first time.

As mentioned in the previous section, a demo program resides in your microcontroller's EEPROM when you receive it. This demo program is written in Motorola's Free-

teract with it. First, make sure you have set up the baud rate, etc., as required for the demo program. Then, switch your board to RUN, and press RESET. Pressing ENTER will cause the menu of the demo program to be displayed. Select the menu commands, as you desire. When you have finished using the demo program, you can return to PCBUG11 by hitting ESC. To re-establish communication between PCBUG11 and the 'HC11, reset your board in BOOT mode, and enter:

baud 9600
restart

4.27 Exploring Further with PCBUG11. For information about PCBUG11 commands and their syntax, enter:

help

at the PCBUG11 prompt. For complete details on PCBUG11, refer to the PCBUG11 Manual. The introductory pages of an HTML version of the manual are included on the Starter Package disk. To access the entire manual in HTML form with your web browser, follow the PCBUG11 link provided on the RESOURCE page of our website (see back cover for our URL).

4.4 Using SBASIC

Here are some example configurations and the approprate /c and /v compiler options used to specify the starting addresses for code (EEPROM, usually) and variables (RAM), where infile is your SBASIC program filename, and outfile is the name you want the target assembly language file to be called.

1) 68HC811E2 in single-chip or expanded mode, with no external E(E)PROM:

sbasic infile /cf800 /v0000 >outfile.asc

2) Adapt11C24DX with 68HC11E0 and 8K EEPROM:

sbasic infile /ce000 /v0000 >outfile.asc

the CONFIG register contents. There is one caveat, however. The new value of the CONFIG register will not take effect until fol lowing a RESET of the 'HC11. Therefore, PCBUG11 will return a BAD MEMORY error when you try to modify the location. This is because the Memory Set and Memory Modify commands automatically attempt to verify the change they just made. In the case of the CONFIG register, reading location $103f returns the contents of a register which reflects the value of the CONFIG register at the time of RESET. It is a read-only register. The actual EEPROM cell implementing the CONFIG register, is a Write-only location, from the user's point of view. Also note: if you write a new 4K mapping address in the CONFIG register of an '811E2, you will not be able to read it with PCBUG11. This is because PCBUG11 must always start up in BOOT mode, which is a single-chip mode. The '811E2 forces the upper 4 bits of the CONFIG register to ones during reset in any single-chip mode. It also forces EEON to one, to ensure that the 2K EEPROM is located in the HC11's vector space, and is enabled. This fact makes it impossible to access locations in the upper 2K of external memory in BOOT mode on a system using an '811E2.

4.26 Using PCBUG11's Terminal Window. PCBUG11 has a basic communications terminal which is useful for testing and debugging your software. To use it, set up the necessary communications parameters such as baud rate, parity, etc. By entering:

control

A list of parameters is displayed. Use the Page Down and Page Up keys to move to the parameter that needs changing. Use the Up and Down arrow keys to make the changes, or enter new values via the keyboard. When you're done, press the ESC key to return to the PCBUG11 prompt. Now enter:

term

to open the terminal window. If you have the Demo program loaded in your board, you can use the terminal window to in-

ware AS11 cross-assembler syntax, and is intended to provide you with an easy way to verify your hardware setup (ie. power supply, serial connection, PC software, etc.). It also provides you with an excellent starting point for developing your own program. Rather than starting from scratch, you can make a copy of the demo source file and remove and add features, to transform it into what you need.

Many people approach programming by spending hours or even days writing a program from scratch, then assembling it and downloading it. Then they cross their fingers and reset the board, praying everything will work. About 99% of the time, their hopes are dashed, as the board does something completely different than they expected, or worse– it appears to do nothing! At that point, they either give up, or purchase expensive diagnostic equipment, such as logic analyzers and in-circuit emulators to begin the long hard road of diagnosing and correcting their software and/or hardware mistakes.

A much more sensible– and rewarding– approach is to start with something that works, and then add new features incrementally. The modular design of Adapt11 gives you that starting point-- hardware that works, and software that works. Now, if you build on that incrementally, each diagnostic step is small and manageable. And it will probably end up taking a lot less time, and costing a lot less money.

A powerful asset for program development is the serial communications interface (SCI). The SCI gives you a window on what's going on inside the microcontroller. Simple diagnostic messages, placed at strategic points in your evolving program, will be invaluable in debugging your software and hardware.

If you look at the demo source code (**demo.asm** for **Adapt11** or **demo8.asm** for **Adapt11C24DX**), you will see that it consists of an initialization section, one main loop, and

several subroutines and interrupt routines. The main subroutine is called ProcessCommand. Its primary function is to interpret a single-character command that is received via the serial port, and perform the appropriate action for that command. You can easily extend the list of commands within the ProcessCommand subroutine by adding code to recognize and process other single-character commands you define. Then it is simply a matter of writing those subroutines to perform the functions you wish to implement. This software structure gives you a way to independently test each new function, one at a time.

For example, let's suppose you are implementing an autonomous vehicle, using a miniature toy car. You will need to read position and collision sensors, and control the speed and directrion of one or more motors. First write a basic motor-speed control routine, and assign some commands to test it. You could assign "F" for faster (increase the motor speed), and "S" for slower (to decrease motor speed). Then implement a pulse-width modulation scheme in your motor speed routine, decreasing the pulse-width by some increment every time the "S" key is pressed, and incrementing pulse-width by the same amount every time the "F" key is pressed. Of course you will want to put tests for minimum and maximum pulse-widths before you decrement or increment, so that the motor doesn't suddenly jump from stopped to full speed when the pulse-width value "wraps" around.

Note: If you plan to incorporate all or part of the demo program in your own code, and you're using a different cross-assembler than AS11, you may have to revise the syntax.

3.21 A Very Simple Program If you are a beginner, the size of the demo program may look overwhelming, and you may

above. To prevent unwanted modification of EEPROM locations, slide the switch back to PROT when you are finished (and before resetting or cycling power). To run the code, switch SW3 away from the SCM position (ie. put it in Expanded Memory mode), switch SW2 to RUN, and press RESET.

It is also possible to use PCBUG11's **loads** command to load an s-record file into external EEPROM, although it is fairly slow. To do this, specify the EEPROM address range, eg.:

eeprom e000 ffff

Then enter:

eeprom erase disable

to disable the Erase-before-write function. Now use **loads**, as described in section 4.22, above.

4.25 Programming the CONFIG Register. The CONFIG register ($103f), allows the user to enable or disble such things as the EEPROM, ROM (or EPROM), and COP (watchdog timer). In the
'811E2, it also specifies to which 4K boundary the internal 2K EEPROM will be mapped *in expanded mode*. The CONFIG register is implemented in EEPROM, so you need to specify this in PCBUG11 before you can modify it. Enter the following command:

eeprom 103f

If PCBUG11 returns a message saying Erase-before-write disabled, enter the following command to enable automatic erase:

eeprom erase enable

Then make sure the PTCON bit in the Block Protect Register is cleared:

ms 1035 0f

Now you can use Memory Modify or Memory Set to change

EEPROM, use the Verify command:

verf \myfiles\myprog

4.23 Accessing External Memory. You can access external memory in any expanded mode system (eg. Adapt11 with MX1 Memory Expansion Card, and all other members of the Adapt11 family) by placing the HC11 in Special Test Mode. (Note: remember that a system must always be in BOOT mode to run PCBUG11; once inside PCBUG11, the mode can be changed by altering the HPRIO register, if necessary). To access external memory, modify the HPRIO register:

ms 103c e5

To change the value stored in a RAM or register location (eg. clear location $2000), simply use the Memory Set command, as follows:

ms 2000 00

To modify an internal EEPROM location, make sure you have first specified the address range of EEPROM, and that you have cleared the Block Protect Register (see Loading an S-record File, above). You can view locations, and modify them as you require, by using the Memory Modify command, as follows:

mm 2000

In this example, PCBUG11 will display the current value of location $2000. If you wish to change it, type the new value, and press ENTER. If not, just press ENTER. The next memory location will be displayed. To end the Memory Modify mode, press ESC on your keyboard.

4.24 Programming External EEPROM. To use the MX1E Memory Expansion Card (eg. **Adapt11 64K Starter Package**) with PCBUG11, slide switch SW1 on the MX1E card to WRITE, and then use Memory Modify or Memory Set, as

wonder what is really required just to do something very simple. In reality, you can throw away almost everything, ending up with about a half dozen lines of code, to produce a very simple program. Rule #1: Every program requires a Reset Vector (located at $fffe and $ffff), since the contents of these locations is loaded into the Program Counter immediately after reset. Therefore, the Reset Vector should point to the beginning of your code.

Below is an example of a very basic program, which just turns on the LED on PORTA (PA6), and then waits in an infinite loop. It will work on any of the Adapt11 family of modules, but was written specifically for the Adapt11 Starter Package (with 68HC811E2). If you are using a different board, you can change the code origin (ORG) to point to your EEPROM start address, but it's not necessary.

```
        org     $f800   ;start of EEPROM in 68HC811E2

begin:  ldaa    #$40    ;write a logic high to PA6
        staa    $1000   ; (and logic low to all other portA bits)
        bra     *       ;branch forever (until reset occurs)

        org     $fffe   ;define the reset vector to point to
        fdb     begin   ;  the beginning of your code
```

Now let's take it one more step: blink the LED on and off. To make the blinking slow enough to perceive with the human eye, we will write a subroutine to create a half-second delay.

Rule #2: *If you are using interrupts, or incorporating any subroutines in your program, you must initialize the stack pointer at the beginning of your program.*

```
        org     $f800   ;start of EEPROM
```

```
begin:  lds     #$ff    ;initialize the stack pointer
loop:   ldaa    #$40    ;write a logic high to PA6
        staa    $1000   ; (and a logic low to all other portA bits)
        bsr     Delay   ;do a time delay
        clr     $1000   ;make all portA bits logic low
        bsr     Delay
        bra     loop    ;do it all again

Delay:
        ldy     #$ffff
D1:     dey             ;pad delay loop with extra cycles
        iny             ;  for total of 15 cycles
        dey             ;this gives approx. 1/2 second at 8MHz
        bne     D1
        rts

        org     $fffe   ;define the reset vector to point to
        fdb     begin   ;  the beginning of your code
```

3.3 Downloading Your Code

Once you have assembled your code with no errors, you can download the resulting s-record file (*filename*.s19) to your board using our Windows-based MicroLoad, or in DOS, using the appropriate batch file provided. Connect the supplied serial cable (handshaking signals are jumpered inside the cable for use with DOS) between connector J2 on your module and COM1 or COM2 of your PC. (You can use a different COM port, but you will need to edit the batchfile to reflect the COM port number you use).

For a single-chip mode system using 68HC811E2 (ie. Adapt11 Starter Package), use **ps1.bat** (for COM1) or **ps2.**

SET in BOOT mode):

pcbug11 -a port=2

(if you're using COM1, omit the **port=2** parameter).

For 68HC11E0, 'E1, 'E9, or '711E9, enter the following command instead:

pcbug11 -e port=2

(if you're using COM1, omit the **port=2** parameter).

If you prefer to use hexadecimal numbers in your PCBUG11 session (instead of the default decimal notation), enter the following command at the PCBUG11 prompt:

control base hex

4.22 Loading an S-record File into 68HC811E2 EEPROM. First tell PCBUG11 the range of addresses that contain EEPROM, as follows:

eeprom f800 ffff

Then make sure the Block Protect Register EEPROM bits are cleared:

ms 1035 10

To do a bulk erase of the entire EEPROM block before programming it:

eeprom erase bulk f800

Then, to make downloading faster, you can disable Erase-before-Write to prevent PCBUG11 from erasing each byte before writing the new value:

eeprom erase disable

Now you're ready to load your s-record file. Suppose you have generated a file called myprog.s19, use the following command to load it into EEPROM:

loads myprog

If it is in a different directory, (eg. c:\myfiles) you should specify the directory path. For example:

loads \myfiles\myprog

To verify that your file myprog.s19 was actually written to

4.2 Using PCBUG11 with Adapt11

Motorola's PCBUG11 is a flexible, powerful, and easy-to-use program that runs on a PC. It allows you to examine and modify on-chip memory (RAM, EEPROM, and EPROM), load HC11 code, debug, trace, erase EEPROM, disassemble blocks of code, assemble line-by-line, and even includes a basic terminal program. What's more, it works with all varieties of HC11 micros, and is fully compatible with the entire Adapt11 family of modules. PCBUG11 is included on your Starter Package disk, and is also available on the Resource page of our website. It is a self-extracting archive, so just copy it to the directory on your harddrive where you want it to reside, and type **pcbug342** at the DOS prompt to extract all the files. If you get a RUNTIME ERROR when attempting to execute this program, check our Tech Support webpage for a patch that resolves this problem.

4.21 Running PCBUG11. Always RESET your module in BOOT mode before starting PCBUG11 (if you're using an **Adapt11** module, also make sure SW3 is set to SCM). It is important to make sure there are no background tasks running on you PC, such as faxmodem drivers, networks, etc. PCBUG11 needs to access the serial port chip directly, and does not co-exist very happily with other programs. Whenever possible, start PCBUG11 directly from DOS (not a DOS shell). If you can't get it to work at all, visit Motorola's freeware website, and look for Application Snapshots dealing with PCBUG11 for more tips on making it work.

It is also strongly recommended that you get the Adobe format version of the Motorola's PCBUG11 User Manual. Look for the link on our website RESOURCE page.

If your micro is a 68HC811E2 (or any A-series chip), type the following command at the DOS prompt (after RE-

bat (for COM2). For expanded mode systems (ie. all other models) , use **xload1.bat** (for COM1) or **xload2.bat** (for COM2). If required, use a text editor to modify these batch files to suit your needs. On a Windows95 system, you may have to add "\dev\" to every reference to the com port path (eg. copy %1.s19 \dev\com1). Instead of a DOS batchfile, you may choose to use Motorola's PCBUG11 program (for DOS) or HCLOAD (for W95/NT) to download your code. See section 4.2 for details.

You may find using the batchfile more convenient if you put it into your working directory, along with the assembler or compiler you are using, and the source code you are writing. Make sure to put the associated boot file in the directory as well (**b8s19bin** is required by **ps1.bat** and **ps2.bat**, while **xload.bin** is required by **xload1.bat** and **xload2.bat**).

Always reset the board in BOOT mode just before you download. Also, if your version of board has Write Protection, make sure to slide the WRITE PROT switch to WRITE. After downloading, switch back to PROT, and switch back to RUN mode. Press RESET to start your code running..

Note: avoid powering up the board with the WRITE PROT switch in the WRITE position. If you do so, one or more external EEPROM locations may get inadvertently corrupted. If this happens, your code may not work as intended, and you'll have to repeat the above procedure to download it again. This precaution only applies to external EEPROM, not to 68HC11 internal EEPROM.

3.4 Using Adapt11 in Expanded Multiplexed Mode

On Adapt11C24DX boards the 68HC11 runs in expanded mode already, since the program memory is in an external chip (EEPROM or RAM). Users of these boards can skip to section 3.5.

Adapt11, however, normally runs in single-chip mode (SCM). To run Adapt11 in expanded mode (with a memory expansion card, for example), slide the SCM switch down. You'll need to switch it back to SCM every time you download code, though so that the MODA and MODB lines of the MCU will be properly configured for Special Bootstrap mode. Note: after changing the SCM switch position, always press RESET to cause the new mode to be activated.

When in expanded mode, the MCU uses PORTC lines to multiplex the 8-bit data bus with the 8 low-order address lines (A0 through A7) and they are referred to as AD0 through AD7. You can use the AS signal with an external 8-bit latch (eg. 74HC373 or 74HC573) to de-multiplex the signals. The upper 8 address bits are generated on PORTB output lines. Control signals E and R/W are then used with address decoding chips, and EPROM, RAM, ROM, EEPROM, or data latches plugged into your solderless breadboard. Note that, even in expanded mode, internally-mapped memory blocks such as RAM, registers, and EEPROM take priority over any external devices mapped into this address space.

3.5 Remapping Internal RAM and Register Block

In some situations, you may wish to move the internal RAM and/or Register Block to a different 4K address boundary. To do this, change the value of the INIT register in the first few lines of your code. Refer to the 68HC11 Programming Reference Guide for information on the INIT reg-

initializes the serial port (SCI), and waits in a loop checking its receive buffer until it gets a value of $FF (BREAK control character). It uses that character to determine the incoming baudrate, and adjusts its BAUD register accordingly. It then places each byte it receives in RAM, storing them sequentially, starting at address 0 and continuing to the end of internal RAM (address $FF on 68HC811E2) For 'HC11 MCUs having more than 256 bytes of internal RAM, loading continues until a timeout delay has elapsed with no characters arriving at the serial port. At this point, the internal bootstrap program loads the Program Counter with 0, causing the CPU to begin executing the program just loaded into RAM. Thus, the program that you have just downloaded begins executing. This feature of the 68HC11 allows an EEPROM erase/write routine to be placed in RAM and executed automatically. Routines for programming internal '811E2 EEPROM (in single-chip mode) and external EEPROM/RAM (in expanded mode) from a Motorola s-record file are included on the Starter Package disk. The source code files are called **b8s19.asm** and **xload.asm**, respectively.

To program the EEPROM in either of these configurations, use the appropriate batchfiles. For '811E2 in single-chip mode (ie. Adapt11 Starter Package), use **ps1.bat** with COM1, or **ps2.bat** when using COM2. For example, to download your s-record file called **mycode.s19** via COM1 to the EEPROM in your Adapt11's 68HC811E2, at the DOS prompt you would enter:

ps1 mycode

For all other Adapt11 versions, you would use **xload1.bat** (for COM1) or **xload2.bat** (for COM2). For example, to download your S-record file called **myprog.s19** to Adapt11C24DX, via COM2, at the DOS prompt, you would enter:

xload2 myprog

4 REFERENCE

4.1 How EEPROM is Programmed

All Adapt11 family modules use EEPROM for program storage (except the EVB versions, which use RAM), and use the MCU in special bootstrap mode to erase and load the code via any serial port. This means you can erase and re-program your code right in-circuit, without the need for special programming boards and UV erasers. This approach results in a low-cost, easy-to-use configuration suitable for education and new product development. Also, there is nothing proprietary about the design configuration and the parts used. The MCU is a standard Motorola part-- you can buy it from Technological Arts or any Motorola supplier to use in your application, without being concerned about availability and licensing.

While most versions of Adapt11 contain EEPROM, the way in which it is programmed is different, depending on whether the EEPROM is inside the microcontroller (eg. 68HC811E2), or in external parallel EEPROM (eg. 28C64 or 28C256). Thus, two different 'boot' programs are provided for programming each type. The concept of the 'boot' program is described below.

All 68HC11 MCUs contain an internal bootstrap loader program in ROM. This program runs whenever the chip is powered up or reset with the mode select pins configured for SPECIAL BOOTSTRAP mode (see Motorola 68HC11 Reference Manual for details). This configuration can be selected on Adapt11 modules by placing the BOOT/RUN switch in the BOOT position. When the RESET button is pressed, the micro executes the internal bootstrap program (bootloader ROM source code listings can be found in Appendix B of the M68HC11 Reference Manual). It first

isters. Take care when relocating RAM, since it is used for the stack. This is especially important if you're using ImageCraft's ICC11 C compiler, since it jumps to your startup code by executing a JSR instruction. If your startup code moves RAM, then the subsequent RTS instruction will not find the return address on the stack (because it won't even find the stack).

3.6 Using a Precision Voltage Reference

Adapt11 family boards use the 5VDC you supply via your breadboard or the on-board 5-Volt regulator (if you supply external power via connector J1) as the voltage reference (VRH) for the analog-to-digital converter on the MCU. If you wish to use a precision voltage reference, you may insert an LM385 or LM336 or equivalent part in location U2. Make sure it is a TO-92 style package. Remove jumper W1 only if you wish to use a VRL other than system ground. It is easiest to use a 2.5V version, but you'll have to make sure none of your analog input voltages will exceed 2.5V, or you risk burning them out. Some 68HC11 users claim that the Vref voltage cannot be as low as 2.5V. However, a careful reading of the Analog-to-Digital Converter Characteristics in the M68HC11 E Series Technical Data book, published by Motorola will debunk that myth. A 2.5V reference is used in many real-world 68HC11-based products, and is perfectly acceptable.

If you wish to use a 5.0V reference, you will need to supply an adequate voltage source for it (above 5V). One possibility is the external voltage supply (if regulated) that you may already be providing to the board via J1. Just desolder the pin 1 end (square pad) of shunt resistor R2, and jumper it to the + input (pin 1) of J1, using a small piece of insulated tubing or heatshrink to insulate the connection.

CAUTION: When using the analog inputs, make sure the input signals do not exceed the reference voltage or go below VRL (GROUND), or damage may occur to the A/D converter.

3.7 About the On-board Voltage Regulator

Adapt11 family boards are supplied with an on-board regulator in a TO-92 package, capable of dissipating about 500 mW at room temperature. Depending on which board you have purchased, you may have either a 'garden-variety' 78L05, or a low-dropout LM2931Z-5. To determine which one you have, look at the part U4 on your board. There are a few differences between the regulators that are worth pointing out.

The 78L05 has a higher quiescent current (several milliamperes), and it will drop out when the input voltage dips below about 8 Volts. Both these factors mean the 78L05 is not well-suited for battery-operated use. Also, the higher the input voltage, the more power the 78L05 has to dissipate in dropping the voltage down to 5 Volts. This means that a high input supply (say 24 Volts) will cause the regulator to go into "thermal shutdown" if you try to draw any additional current from the regulator than what is already being drawn by the circuitry on board. Another drawback to this regulator is this: if you decide to provide an external regulated 5 Volt supply via the 50-pin header H1, the 5 Volts will feed into the output of the on-board 78L05, and it will draw several milliamps, adding to the load on your power supply. It could also damage the 78L05 if the input side (ie. the + pin of J1) were inadvertently grounded. The effect would be reverse voltage on the 78L05.

The LM2931Z-5, has some nice features, such as a very low quiescent current, and will work with an input voltage

down to 5 Volts (or below), making it quite well-suited to battery operation. It is also designed to withstand reverse polarity and, if unused, does not present a load to an external regulated 5-Volt supply applied via the 50-pin header H1. One drawback, however, is that it can become unstable and start to oscillate at low temperatures, especially if the input voltage source is connected to J1 via long wires. In the former circumstance, the on-board 10uF tantalum capacitor can be replaced with a higher value (47uF or 100uF). To compensate for long lead-in wires, add capacitance of 100uF or so at, or close to, the J1 connector.

Most of the Adapt11 family circuit boards were designed to accomodate a voltage regulator with a tab-style package, to provide more current at 5 Volts for your application circuitry. You can remove the TO-92 package and replace it with a TO-220 package if necessary. The double line shown on the board outline shows how the metal tab of the package should be oriented. Remember that the package tab is Ground, so be careful not to let it short to other components on the board. Another possible mounting method is to insert the voltage regulator pins through the *solder side* of the board, and bend the package body parallel to the plane of the board. This will place the tab away from the board, and it can be safely attached to a heatsink, if desired. Double-check the orientation of the new regulator with the manufacturer's data sheet and the board's schematic, keeping in mind that square pads on the board always designate Pin 1.