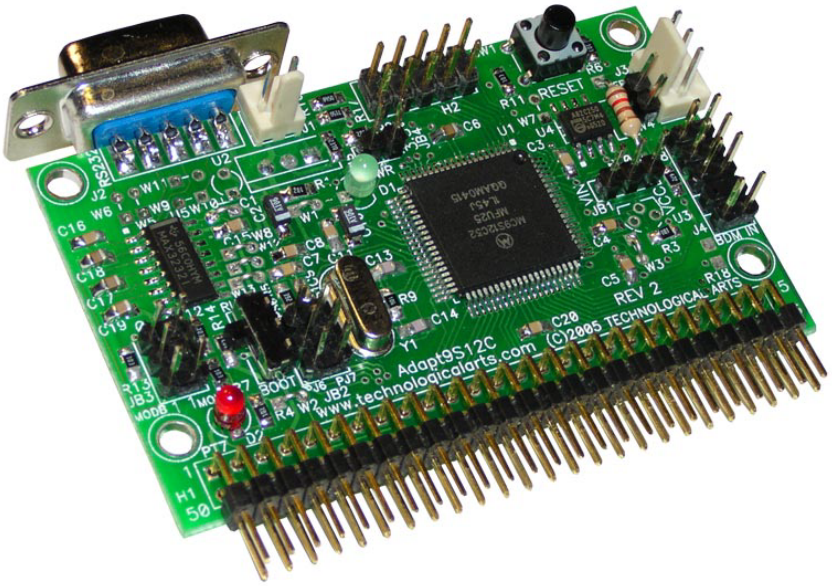
Using Your Adapt9S12C Microcontroller Module



Technological Arts

www.technologicalarts.com

Revison 0b

DISCLAIMER

While we have made every effort to avoid errors in the preparation of this manual, we cannot be held responsible for any misinformation or omissions that may have occurred. Furthermore, as manufacturer of this product, **Technological Arts**' sole liability and the buyer's exclusive remedy shall be refund of the amount paid or repair or replacement of the product, at the manufacturer's option. The manufacturer disclaims all other warranties, expressed or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the product and accompanying written material, hardware, and firmware. In no event shall the manufacturer or its suppliers be held liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other loss) arising out of the use of, or inability to use, the product, even if the manufacturer has been advised of the possibility of such damages. The product is not designed, intended, nor authorized for use in applications in which the failure of the product could bring about a scenario in which personal injury or death may occur. If used in any such unintended or unauthorized application, the manufacturer and its suppliers shall be held harmless against all claims, even if any such claim alleges that the manufacturer was negligent regarding the design or implementation of the product.

Product features, availability, and prices may change without notice.

E S D W A R N I N G

This product, like all microcontroller products, uses semiconductors that can be damaged by electrostatic discharge (ESD). When handling, care must be taken so that the devices are not damaged. Damage due to inappropriate handling is not covered by the warranty.

The following precautions must be taken:

- Do not open the protective conductive packaging until you have read the following, and are at an approved anti-static work station.
- Use a conductive wrist strap attached to a good earth ground.
- If working on a prototyping board, use a soldering iron or station that is marked as ESD-safe. Always disconnect the microcontroller from the prototyping board when it is being worked on.
- Always discharge yourself by touching a grounded bare metal surface or approved anti-static mat before picking up an ESD-sensitive electronic component.
- Use an approved anti-static mat to cover your work surface.

1 INTRODUCTION

1.1 WELCOME!

With **Adapt9S12C**, you are now ready to explore the power and versatility of Freescale's advanced 16-bit microcontroller family! Whether you're new to Freescale microcontrollers or you've used some of the earlier ones (such as 68HC05, 68HC11, or 68HC12), you'll be impressed with the well thought-out design and implementation of the HCS12 family. **Adapt9S12C** gives you the opportunity to explore the 9S12C family's potential at a very affordable price! Add to that the proven advantages of it's popular DIP format, and you'll see why you picked a real winner!

1.2 SUPPORT

To help you get the most out of this product, and to make the experience as enjoyable and productive as possible, we've put together a comprehensive website, loaded with resources, support, and applications information. If you experience any difficulties, or need help with your application, the World Wide Web is arguably the most valuable resource available to you. There you'll find the latest information, software, and trouble-shooting help, as well as discussion groups where you can network with people around the globe to get the answers you need. So if you still need help, or have questions after reading this manual and perusing the contents of the included CD, visit our Support Forum: **www.technologicalarts.net** and tap into the collective! Also, be sure to join Freescale's **16-bit Microcontroller** discussion group, at http://forums.freescale.com/freescale/board?board.id=16BITCOMM

1.3 PRODUCT CONFIGURATION

Adapt9S12C includes all the essential support circuitry that 90% of applications use (e.g. crystal, voltage regulator, reset switch, RS232 interface), without tying up port pins with costly interface circuits that most users won't need (e.g. clock/calendar chip, serial EEPROM). It is form-factor compatible with our original 68HC11 product, the immensely-popular Adapt11, and comes with the legendary "SB" connector option (a Technological Arts innovation), enabling it to be plugged directly into a solderless breadboard-- just like a big chip. In fact, it is offered with more than a dozen connector options, making a vast range of configurations possible when coupled with the complete selection of prototyping cards, backplanes, and application cards that are available from Technological Arts. From MCU evaluation, training, product development, hobby or school projects, and semi-custom solutions all the way to being embedded right into a commercial product or system, Adapt9S12C is a very cost-effective product. The board is currently offered in two versions, which differ only in memory sizes:

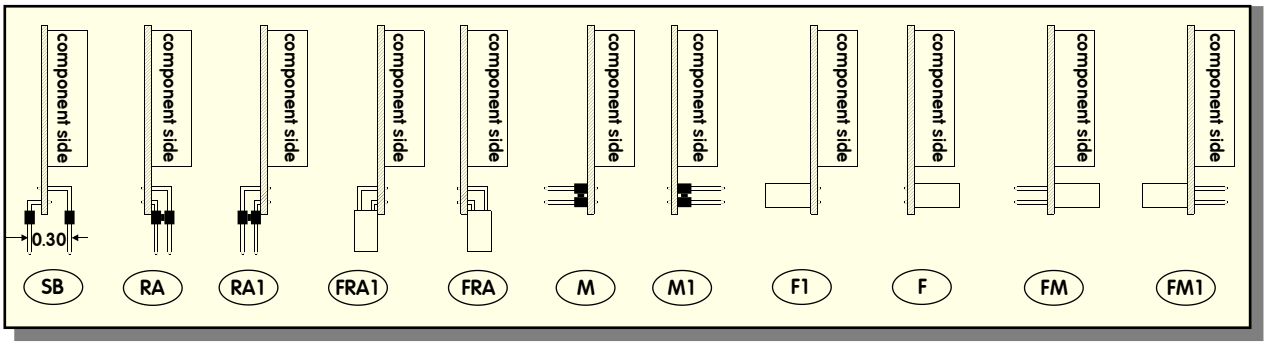
- Adapt9S12C32 has 32K Flash (program memory) and 2K RAM (data memory)
- Adapt9S12C128 has 128K Flash (program memory) and 4K RAM (data memory)

1.4 ADAPT9S12C VS. TRADITIONAL EVALUATION BOARDS

Most available evaluation and development systems tend to be too expensive and bulky for embedding into a real application, so they lie on a shelf gathering dust once you've reached a certain point in the learning curve. Or maybe you think up some clever way to hack it apart and make it fit inside your robot or product prototype. Even then, the prototyping area provided is often limited, and does not lend itself to re-usability. And what if you burn out a chip the night before the contest or product demo? What a mess to repair or re-design!

Adapt9S12C solves all of these problems and more! Since it brings out practically

Standard Connector Options (use NC for "no connector")

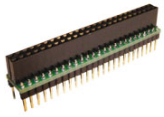


all I/O lines and control signals to a standard 50-pin interface connector (H1) and 10-pin auxilliary connector (H2), it is modular and re-usable. Interface cards can be unplugged and upgraded, or the whole system can be re-configured at the last minute. What's more, the program memory can be erased and re-programmed in seconds, right in place. The multitude of connector options enables you to use the module in whatever way best suits your application-- board-stacking, end-to-end (planar), backplane, ribbon cables, etc. A full range of accessories including backplanes, prototyping cards, and application-specific cards is available, and more accessories are being developed all the time. Make a point to visit our website from time to time, just to see what's new.

When used with the RA1 connector option, a prototyping card, experimentor card, demo card, or motor driver card can be plugged directly onto the I/O connector, forming a planar arrangement. Advantages of this configuration are easy access to all I/O pins for probing and measurement, and easy prototyping of your interface circuits. What's more, the detachable nature of the cards means that you can easily replace them with other cards. You can build up a collection of different application circuits, and use them all with the same microcontroller board. This is especially advantageous in an educational environment, where the student can progress from simple to more complex applications throughout a semester, or from one course to the next-- perhaps even incorporating the board into a final project. In fact, where budgets are tight, different students can share the same microcontroller module, and plug in their own interface cards when it's their turn to use it.

1.5 USING ADAPT9S12C WITH SOLDERLESS BREADBOARDS

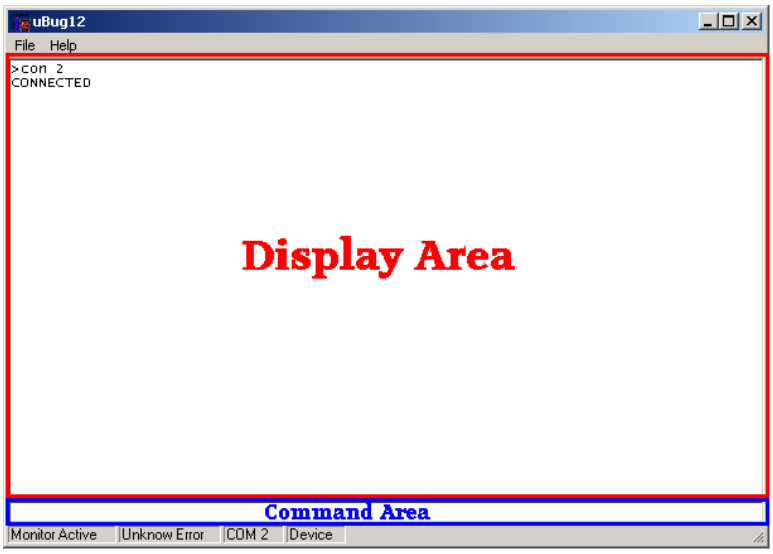
When used with the SB connector option (or optional adapter, #ADHDR50-F), the module will easily plug vertically into any standard solderless breadboard. The resulting footprint is equivalent to a 50-pin narrow DIP, and has a similar pin-numbering sequence (ie. wraps around the end, from pin 25 to 26). Plug the adapter into your breadboard, wire up your circuits, as required, and then plug the module into the adapter. If you want to access signals on H2 as well, you'll need a 10-pin ribbon cable and a breadboard adapter (#ADIDC10-M) to bring those signals down to the breadboard. Note that pin-numbering of the 10-pin connector is like a typical ribbon cable connector, with odd-numbered pins on one row and even-numbered pins on the other.



Another popular breadboarding approach is a planar configuration utilizing a card mounted with a solderless "experimentor" board that plugs into H1. This card (#AD12EXPH1-FRA1) features a 50-pin dual-row receptacle ("F"-style connector) next to the breadboard section, giving you easy access to all of the I/O signals. To build your circuit, it's just a matter of plugging lengths of ordinary #22 hookup wire between the signals on the receptacle and the places you need them on the breadboard.

1.6 RESIDENT DEBUG/MONITOR

Residing in a 2K protected block of on-chip flash memory is Freescale's versatile Serial Monitor program. When used with uBug12 (a free Windows application created by Technological Arts), you can display and edit memory and registers, erase and program flash, set breakpoints, and do instruction tracing. Flip a switch on the board to Run mode, and your program runs automatically from Flash, following reset or powerup. See Chapter 2 and Appendix A for details on uBug12 and the resident monitor.



1.7 COMMUNICATIONS

An RS-232C serial interface port connector (9-pin D-sub) is included, enabling communication with a PC com port, or any other device which has an RS-232 serial port, via a standard 9-pin serial port extension cable. The RS-232 channel is implemented via the SCI of the MCU, and when the board is reset in Load mode, the resident Serial Monitor uses this port to communicate with an appropriate program running on your PC (e.g. UBug12, CodeWarrior, or NoICE12). In Run mode, the RS-232 port is available for your application.

The module includes physical layer support for Controller Area Network (CAN), with CANHI, GROUND, and CANLO signals brought out to J3. A mating 3-pin Molex connector is available from Technological Arts (#HPCT3). A 120-Ohm terminating resistor is included on the board, and can be activated by shorting link W4.

In passing, it should be mentioned that the MCU also supports Serial Peripheral Interface (SPI). Since this is a logic-level protocol, meant for local communications among peripheral chips, no transceivers are required nor are they provided. Commonly used SPI chips and modules include: serial memory (e.g. EEPROM, Flash), temperature controllers, clock/calendar chips, DACs, MP3 decoders, etc.

See the 9S12C datasheets for details on all of these subsystems.

1.9 JUMPER OPTIONS

The user has a choice of selecting PJ6 and PJ7 or PS0 and PS1 to be brought out to the I/O connector (H1). The selection is implemented via jumper block JB2. By bringing out PS0 and PS1, the SCI can be accessed at the logic level, and is useful for such applications as GPS and serial LCD interfacing.

MCU Mode Select jumper block JB3 provides access to the microcontroller's MODA and MODB pins, for implementing single-chip mode, narrow expanded mode, and wide expanded mode. The default setting is Single-chip Mode (MODA=MODB=0). Refer to the 9S12C data sheet for details on implementing other modes.

2 GETTING STARTED

- ***Be sure to read and follow the Safe Handling Procedures outlined inside the manual's front cover***
- Perform a visual check of the hardware for any damage during transit
- Connect the RS232 port to a com port of a personal computer, using the cable supplied
- Locate and install the Windows application called uBug12, included on the CD (also on our website)
- Launch uBug12
- Activate the serial port connection by entering CON x (where x is the comport you are using; usually 1 or 2)
- Set switch SW2 to the LOAD position
- Connect the supplied power source to J1
- uBug12 will display a short message, followed by its command prompt
- The Module is now ready to accept your commands (see Appendix B for a full list of uBug12 commands)

To download one of the supplied example programs into flash and execute it, follow these steps:

- Type **fbulk** <enter> at the uBug12 prompt to erase any existing program
- Type **fload** <enter>
- From the displayed file browser, select one of the example program's output files (.s19 or .s28 file)
- After loading has finished, move SW2 to the RUN postion, and press the Reset button (SW1)
- The program will run
- If you wish to debug the program, move SW2 back to the LOAD position and press Reset
- The uBug12 prompt will appear
- Use uBug12 commands to debug your code

2.1 POWER OPTIONS

A DC power supply is included with most bundle configurations. It is the recommended power source when you're starting out. If for some reason it's not convenient (eg. you don't want an extension cord trailing around behind your robot :>), there are a couple of alternatives:

Option 1: connect a DC voltage of 6 Volts or more (maximum: 24 VDC) via the external power connector, J1. Your DC supply does not need to be regulated, but it should be capable of supplying at least 100 mA (more if you will be using a BDM pod, or you're driving other circuits as well). If your supply will also be driving motors, make sure to isolate it before feeding it into the module (to protect it from electrical noise generated by the motor coils). You can do this by putting inductors (10uH, nominal) in series with both the + and - leads. Preferably, use the red & black power wire provided (order code: PCJ1-8). Red is positive, and black is negative (ground). *CAUTION! Make sure you have the polarity correct!*

Option 2: supply regulated 5VDC (or 3V if using 3V mode) via the appropriate pins on the module. See module pinout diagram in Appendix C for the Vcc and GROUND pins to use. *CAUTION! Leave J1 unconnected. Double-check your connections before applying power! If you are applying 3V, be sure to read the notes on 3-Volt operation in chapter 3.*

2.2 DEMO PROGRAMS

We've included a few demo programs on the CD-ROM (also available from the product webpage) to give you a starting point. There are some examples in C and some in assembler. The source code is included, so you're free to modify them all you want! Visit our Support Library and the product webpage (once there, click on the Resources tab) for more examples, application notes, and links to third-party sites.

2.3 USING THE DEBUG/MONITOR

uBug12 is a Windows-based graphical user interface (GUI) for Freescale's HCS12 Serial Monitor program. It aims to emulate the most common debug/monitor commands, and to provide an easy-to-use interface. The following paragraphs will help you get started with uBug12.

Install uBug12 from the included CD simply by unzipping it onto your C: drive. Launch the application, and then apply power to the module. A sign-on message will appear in the uBug12 window. A full list of uBug12 commands is provided in Appendix A.

The first step after you have launched uBug12 is to activate the connection to the target device (ie. The MCU module) which is running the Serial Monitor Program. This is done via the CON command, which takes one parameter: the number of the comport to which the device is connected.

eg. **CON 2**

In this example, uBug12 will open a connection to the target hardware attached to com2. Then you can choose any of the uBug12 commands.

One of the most useful commands is FLOAD. This command lets you load an S-record file into the target device flash. If you are loading a file containing linear S2 records, just type FLOAD, if you are using a banked S2, such as those generated by ImageCraft's ICC12, Metrowerks Codewarrior, and other tools, use FLOAD ;B. Once you have typed the FLOAD command followed by the ENTER key, a dialog box will appear and you will be able to browse to the file you want to load. If you have omitted the ;B parameter, only files with an S2 extension will be displayed for loading. Before loading another file, though, make sure to erase flash, using the FBULK command.

One nice feature you'll discover is command line history. Use the UP and DOWN arrows on your keyboard to recall previously typed commands. You can then edit them, as needed, before hitting <ENTER>.

2.4 NEW TO FREESCALE MICROCONTROLLERS?

If you've come from an 8051, PIC, or other background (or have never used microcontrollers before), you should get up to speed on Freescale MCUs by reading *Understanding Small Microcontrollers,* found on the CD-ROM. Written by Freescale's Jim Sibigtroth, principal design engineer of the HC12 family, this excellent book uses an earlier MCU (68HC05) to introduce you to the basic concepts and design philosophy upon which the 9S12 was built. You should also make sure to have a copy of the MC68HC11 Reference Manual, since it contains detailed descriptions and examples for many of the hardware subsystems.

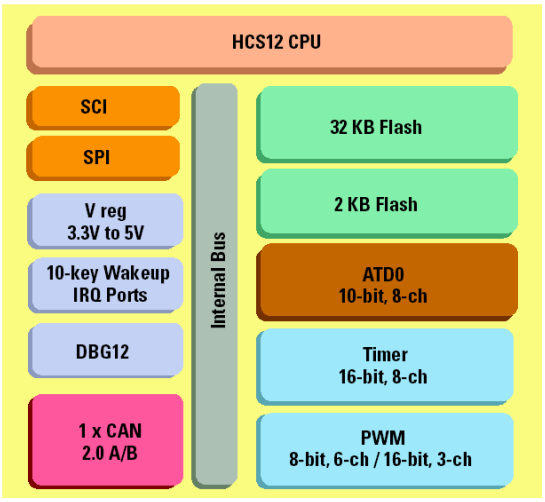
2.5 MIGRATING FROM 68HC11

If you are already experienced with the 68HC11 family of microcontrollers, writing programs for the HCS12 family will not present a big challenge (don't throw away your HC11 Reference Manual-- the trusty "pink book"). In fact, you can use your existing 68HC11 assembly code and re-assemble it to run on the CPU12 core, but there are a few things to keep in mind.

Assembler syntax. You may need to edit your source file to conform to the syntax and directives of the HC12 assembler you'll be using. There are several assemblers available (e.g. AS12, MiniIDE, IASM12, MCUez), and each has its own syntax to be aware of.

Register Block. Instead of $1000, the register block default location is $0000

through $03FF, and there are a few hundred registers! You'll need to locate the relevant registers for the subsystems you plan to use, and make sure they are properly configured.



RAM location. Following reset, the memory map configuration has the register block overlapping RAM, starting at $0000, with registers taking priority, so the first 1K bytes of RAM are not usable. In order to free up all of the RAM. the monitor program re-maps RAM to start at $3800 through $3FFF (via the RAMinit register). This means you'll need to initialize the Stack Pointer to $4000 (on the HCS12, the stack pointer points to the address following the top of the stack).

High-speed Bus. The default bus speed is half the crystal frequency of 8 MHz, so it is 4 MHz. If you enable the PLL, it will be even higher (up to 24 MHz). This will mean changing some initialization values for control registers and revising delay constants if you are using any software timing loops in your old 68HC11 code.

I/O Ports. The digital I/O ports on the HCS12 are more flexible than ever. Besides selecting the direction of each port pin via a Data Direction Register, there are registers controlling output drive level (standard and reduced), internal pullup and pulldown resistors, and output logic polarity (ie. true or inverted logic).

COP Watchdog. On most flavours of HC11, this could be enabled via a bit in the non-volatile CONFIG register. On the HC12, it is dynamic, and automatically enabled following reset. Therefore you have to choose whether you're going to service it, or disable it.

Write-Once Registers. On the HCS12, there is no 64-cycle startup window in which you have to write all the protected registers. Instead, the HC12 implements a WriteOnce rule on sensitive registers. What this means is that, following reset, you have one chance to write them, then they become "Read Only". The advantage of this is that you have more control of when you alter these register values. To take advantage of this safeguard, you should initialize all the registers that are crucial, even if the default values are what you want. That way, if your code runs amok, or there are any glitches which try to change register values, they will be protected.

There are many more differences, and you should make sure to read through the Freescale App Note (AN1284) that details the new instructions and addressing modes of the 68HC12, explaining differences from the 68HC11.

2.6 MIGRATING FROM THE 68HC912

You gain a lot more speed, memory, and flexibility, but you have a lot more registers to think about, and many of their addresses have changed. Gone are the Vfp generator and flash voltage switch, since the new flash technology uses 5V, and has built-in self-timed algorithms for program and erase functions. But your s-records must contain an even number of bytes, and begin on an even address boundary if you're going to "burn" them into flash. Some assemblers will generate this format for you but others, such as the included AS12, don't. In the latter case, you'll need to use the utility called SRECCVT.

3 HARDWARE DESIGN FEATURES

3.1 3-VOLT OPERATION

One of the nice features of the 9S12C is that it can operate on 3 to 5 V, while maintaining full bus speed capability. To support 3 Volt operation, the module incorporates an adjustable regulator whose output voltage is set by a resistive voltage divider. The circuit has been designed such that simply inserting a shorting jumper causes the regulator's output to shift from 5V to 3.3V. When operated at 3V, there are a few precautions that should be noted, however.

The logic pins are ***not*** 5V-tolerant, so you will need to take the necessary steps to prevent damage to the I/O pins of the MCU. Also, the maximum VRH voltage is limited to 3.3V, so any external voltage or precision voltage reference you supply should be scaled accordingly. One last point is that some BDM pods will not work with 3 Volt targets, so you should check the specs of the BDM pod you intend to use. A good choice is our MicroBDM12LX (#UBDM12LX), which works at both 3V and 5V.

3.2 RESET

Unlike previous HC11 and HC12 designs, the 9S12C MCU has an on-chip low-voltage inhibit (LVI) reset circuit, so it is not necessary to provide such a circuit externally. A momentary tact switch is provided for manual reset, and the LVI circuit will provide a clean reset signal upon power-up.

3.3 ABOUT THE VOLTAGE REGULATOR

Adapt9S12C includes an LM1086CT-ADJ voltage regulator. Housed in a TO-220 package, it is capable of handling a whopping 1.5 Amps at room temperature! Other nice features are: reasonably low quiescent current (5mA, typical), and low dropout voltage (1V @ 1A)-- it will work with an input voltage down to about 6 Volts, making it quite well-suited to battery operation. It is also designed to withstand reverse polarity. One drawback, however, is that it can become unstable and start to oscillate at low temperatures, especially if the input voltage source is connected via long wires. If low-temperature operation is anticipated, 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 at, or close to, connector J1. Refer to the manufacturer's data sheet for more.

Heatsinking. Because the regulator is mounted on the underside of the circuit board, with the package body parallel to the plane of the board, it can be safely attached to a heatsink. Many clip-on heatsinks are available for use with TO-220 packages. Another option is to mount the board on a sheet of aluminum, using standoffs and insulated hardware. If the appropriate length is chosen for the standoffs, the tab of the regulator will lie flush with the aluminum sheet, and can be coated with silicone grease and bolted (or riveted) to the plate (tightening a nut and bolt will require a little ingenuity). ***CAUTION! Unlike some other voltage regulators, the metal tab is not Ground-- it is connected to Vout so, in most cases, you will need a heatsink insulation kit.***

3.4 PLL

While the supplied crystal is only 8MHz, the MCU is capable of running at a much higher speed. The phase-locked loop feature of the MCU allows you to boost the bus speed by an integer multiple of the crystal frequency, so by enabling the PLL, you can actually run the MCU at 24Mhz.

3.5 ADDITIONAL INPUT/OUTPUT PINS

An additional I/O connector is located on the upper edge of the module, providing access to an another 8-pin I/O port (PORTH). The extra two pins on the connector bring out Ground and Vcc for convenience. This means a standard 10-pin ribbon cable may be used to interface them to your circuit. We make a 10-pin solderless breadboard adapter (#ADIDC10-M) which can be used with a 10-conductor ribbon cable (#RC10FF6) to bring the extra pins down to your solderless breadboard.

4 WRITING SOFTWARE

4.1 IMPACT OF THE SERIAL MONTIOR

When you are working without a BDM pod, the Serial Monitor program is the only method available to load and erase flash. It is in a protected block of flash, so there's no way to accidentally erase it. There are two modes, controlled by switch SW2: Run and Load. The monitor mode is determined immediately following reset by checking the position of switch SW2. When working with the monitor program in place, there are a few points to be noted:

1) while the user vectors are implemented by the monitor at 0xF780 to 0xF7FF, you don't really have to worry about it, because the monitor program will automatically adjust them when your s-record is loaded.
2) the monitor relocates RAM to the address range 0x3800 to 0x4000 from the default location after MCU reset of 0x0000 to 0x07FF.
3) the monitor program enables the phase-locked loop (PLL), so the target is running at 24Mhz (when in LOAD mode) and not at the startup speed of 4Mhz.
4) the user code must clear the CCR I-Bit, either via a CLI in assembler or via the INTR_ON() in ICC12.
5) SCI0 cannot be used by the user program when in LOAD mode, since it is dedicated to the monitor program.
6) COP cannot be disabled in Load mode.

4.2 WRITING A SIMPLE C PROGRAM IN ICC12

Before starting, you'll need to set up your compiler settings, as follows:
Program Memory = 0x4000.0x7FFF:0xC000.0xFFFF
Data Memory = 0x3800
Stack Pointer = 0x3FC0

Note that the Data Memory and Stack Pinter addresses shown are valid only for a device with a resident monitor, since the monitor remaps the RAM following reset. If you are writing software for a completely blank chip, and loading it in via a BDM pod, you'll need to change these values to work with the default RAM address range (see the MCU datasheet).

```
//this programflashes LED D1 on PP0 twice a second
#include <hcs12c32.h>

#define DUMMY_ENTRY  (void (*)(void))0xFFFF

#pragma nonpaged_function _startextern void _start(void);/* entry point in crt12.s */
void main(){
```

```
        INTR_ON();            //needed for the SerialMonitor

        DDRP = 0x01;          //Enable LED port

        RTICTL = 0x7F;      //Set RTI divider for 4Hz time base
        CRGFLG |= 0x80;   //Clear the RTI Flag
        CRGINT |= 0x80;   //Enable the RTI
}#pragma interrupt_handler rti_handler
void rti_handler(){
       CRGFLG |= 0x80; // Clear the RTI Flag
       PTP ^= 0x01; //Toggle LED
       INTR_ON(); //Enable Interrupts
}

   #pragma abs_address:0xFFF0

   void (*interrupt_vectors[])(void) =
   {
       rti_handler, /*Real Time Interrupt*/
       DUMMY_ENTRY, /*IRQ*/
       DUMMY_ENTRY, /*XIRQ*/
       DUMMY_ENTRY, /*SWI*/
       DUMMY_ENTRY, /*Unimplemented Intruction Trap*/
       DUMMY_ENTRY, /*COP failure reset*/
       DUMMY_ENTRY, /*Clock monitor fail reset*/
       _start, /*Reset*/
   };
   #pragma interrupt_handler rti_handlervoid rti_handler(){
       CRGFLG |= 0x80; // Clear the RTI Flag
       PTP ^= 0x01; //Toggle LED
       INTR_ON(); //Enable Interrupts
}

   #pragma abs_address:0xFFF0

   void (*interrupt_vectors[])(void) =
   {
       rti_handler, /*Real Time Interrupt*/
       DUMMY_ENTRY, /*IRQ*/
       DUMMY_ENTRY, /*XIRQ*/
       DUMMY_ENTRY, /*SWI*/
       DUMMY_ENTRY, /*Unimplement Intruction Trap*/
       DUMMY_ENTRY, /*COP failure reset*/
       DUMMY_ENTRY, /*Clock monitor fail reset*/
       _start, /*Reset*/
   };
```

4.3 OTHER ISSUES WITH ICC12

Because the register addresses have changed from what they were in HC12, meaning the header file is different for the C32, some library files in ICC12 will need to be re-compiled, using the new header file, if you want to use them. Of course, if you're not using library functions, or you are using functions that don't involve registers, then there

won't be a problem with the existing versions. The modified functions are included on the CD-ROM to get you started.

To use the SCI, make sure to include ***C32_iochar.c*** and ***C32_serial.c***. Also, you'll need the complete vector file for the C32, which is called ***C32_vectors.c***. Unzip ***C32_C.zip*** and place ***hcs12c32.h*** in ***c:\icc\include*** (or in your equivalent path). Make sure to place ***C32_Vectors.c*** in the same folder as your project, and add it to you project via the "add file menu item".

4.4 "HELLO WORLD" PROGRAM

First of all create a new project from the Project menu.

Then create a new file and save it as HelloWorld.c Add it to the Project by right clicking in the Project Panel and using Add Files to add it to the Files section.

Next type in the following code:

```
#define _SCI
#include <hcs12c32.h>

#pragma nonpaged_function _start
extern void _start(void); /* entry point in crt12.s */

extern int _textmode;

int putchar(char c)
     {
     if (_textmode && c == '\n')
            putchar('\r');
     while ((SC0SR1 & TDRE) == 0)
            ;
     SC0DRL = c;
     return c;
     }

void main(){
      INTR_ON();            //need for the SerialMonitor

      DDRP = 0x01;          //Enable LED
     SCI0BD = 26; //9600 Baud
     SCI0CR2 = 0x0C; /* enable transmitter and receiver */

     puts("Hello, World!");

}

#pragma abs_address:0xFFFE

void (*interrupt_vectors[])(void) =
{
     _start, /*Reset*/
};
```

Since **puts** calls **putchar**, we define it before invoking it in **main**. **Main** has an implicit

_Start entry point, which is called after the setup by **CRT12.o**, which is a module that the ICC12 linker links in as the starting point of the program. Besides initializing the stack and other system features it initializes memory, initialized variables and constants before transferring control to the Main.

Compile and link the program, fixing any syntax errors that may have cropped up. Ensure that the Project Options | Device Configuration drop down box points to the 9S12C32 Flash Mode. This sets the link address to start the code section at 0x4000 and the stack at top of RAM (0x4000).

```
        return c;
        }

    void main(){
         INTR_ON();            //need for the SerialMonitor

         DDRP = 0x01;          //Enable LED
        SCI0BD = 26; //9600 Baud
        SCI0CR2 = 0x0C; /* enable transmitter and receiver */

        puts("Hello, World!");

       }

       #pragma abs_address:0xFFFE

       void (*interrupt_vectors[])(void) =
       {
        _start, /*Reset*/
       };
```

Since **puts** calls **putchar**, we define it before invoking it in **main**. **Main** has an implicit **_Start** entry point, which is called after the setup by **CRT12.o**, which is a module that the ICC12 linker links in as the starting point of the program. Besides initializing the stack and other system features it initializes memory, initialized variables and constants before transferring control to the Main.

Compile and link the program, fixing any syntax errors that may have cropped up. Ensure that the Project Options | Device Configuration drop down box points to the 9S12C32 Flash Mode. This sets the link address to start the code section at 0x4000 and the stack at top of RAM (0x4000).

4.5 USING A BDM POD

If you have a BDM pod, you can erase the resident monitor program completely. This will free up all the MCU resources for your program (most importantly, the SCIs). Without the monitor in place, the RAM will be at the default location following reset, so make sure to use the correct compiler/linker settings. Also, the PLL won't be enabled, so the bus speed will be 4 MHz.

4.6 AUTOMATING S-RECORD CONVERSION IN ICC12

You may have to convert the s-record file to get it into the proper format for your BDM pod to load correctly. ICC12 has a nice feature at *Project->Options->Compiler->ExecuteCommandAfterBuild* where you can add the SRECCVT command mentioned earlier.

5 GOING FURTHER

If you'd like to get started interfacing common electronic devices such as LEDs, switches, relays, etc., you may consider purchasing the optional Demo Card (Adapt9S12DemoH1, shown). It includes a light sensor, thermistor, bargraph LED, DIP switch, potentiometer, audio transducer, and a couple of logic MOSFETs, and has support for an optional character LCD.

Several other Application Cards are available, including

• a Display/Keypad/Keyboard Interface (DKKI) which supports character LCDs, PS/2 keyboard, and/or matrix keypads. Communication with the LCD is accomplished via the SPI, using a serial shift register, reducing the number of port pins required.

• a Voice Record/Playback Module (AD11DXVRPM) which incorporates the Winbond ISD2560 60-second solid state record/playback chip, and supports MCU control

• a Servo/Sensor Interface Module (AD9S12SSIM) which supports standard robotics applications (hobby servo control, IR distance-measuring sensors, sonar distance-measuring sensors, audio microphone, audio transducer, etc.)

• a 3-axis Bi-polar Stepper Motor Controller board (AD12DXXYZSM), which can be utilized to implement CNC applications

Check the subcategory called Application Cards on our website to browse the currently available selection.

APPENDIX A - SERIAL MONITOR

INTRODUCTION

This appendix describes the Freescale 2 Kbyte monitor program for the HC9S12 series MCU. This program supports 23 primitive debug commands to allow FLASH / EEPROM programming and debug through an RS232 serial interface to a personal computer. These include commands to reset the target MCU, read or modify memory (including FLASH /EEPROM memory), read or modify CPU registers, go, halt, or trace single instructions. In order to allow a user to specify the address of each interrupt service routine, this monitor redirects interrupt vectors to an unprotected portion of FLASH just below the protected monitor program. This monitor is intended to be device unspecific, this single application with very slight modification should execute on any HC9S12 derivative. A user on a tight budget can evaluate the MCU by writing programs, programming them into the MCU, then debug using only a serial I/O cable and free software (uBug12) for their personal computer.

This monitor does not use any RAM other than the stack itself. The COP watchdog is utilized for a cold reset function; user code should not disable the COP (ie. by writing 0x00 to COPCTL). This development environment assumes you reset to the monitor when you are going to perform debug operations. If your code takes control directly from reset, and then an SCI0 interrupt or a SWI attempts to enter the monitor, the monitor may not function because SCI0, the phase locked loop (PLL), and memory initialization registers may not be initialized as they would be for a cold reset into the monitor. There is no error handling for the PLL. If the frequency source is missing or broken, the monitor will not function. The monitor sets the operating speed of the MCU to 24 MHz. Modification of the MCU speed by the user with out considerations for the monitor program will render the monitor nonfunctional. If the PLL loses lock during operation, the monitor will fail.

BLOCK PROTECTION

In order to prevent accidental changes to the monitor program itself, the 2 Kbyte block of FLASH memory where it resides ($F800-$FFFF), is block protected. Additionally all write commands are restricted from modifying the monitor memory space. The only way to change the contents of this protected block is to use a BDM-based development. In the lowest cost applications where the monitor is used with an SCI serial interface to the RS232 serial port on a personal computer, there is no way to accidentally erase or modify the monitor software.

COP CONFIGURATION

The monitor as written creates hard reset function by using the COP watchdog timer. It does so by enabling the COP and waiting for a COP timeout reset to occur. If the user application uses the COP two issues must be considered.

•If the COP is disabled in the user application, the monitor will be unable to perform a hard reset and will soft reset to the start of the monitor instead.

•The monitor does not service the COP timer. If the user application implements COP timer servicing, upon re-entry into the monitor a hard reset is likely to occur.

MEMORY CONFIGURATION

1) Register space is $0000-$03FF.

2) Flash memory is any address greater than $4000. All paged addresses are assumed to be Flash memory.

3) RAM ends at $3FFF and builds down to the limit of the device's available RAM.

4) External devices attached to the multiplexed external bus interface are not supported.

SERIAL PORT USAGE

In order for this monitor to function the SCI0 serial interface is used. It is assumed that the monitor has exclusive use of this interface. User application code should not implement communications on this serial channel. This monitor accommodates RS232 serial communications through SCI0 at 115.2 kbaud. For applications requiring the use of SCI0, you should purchase a BDM pod which allows for more advanced debugging.

VECTOR REDIRECTION AND INTERRUPT USE

Access to the user vectors is accomplished via a jump table located within the monitor memory space. This table points all interrupt sources to a duplicate vector table located just below the monitor. ($F780-$F7FE). The monitor will automatically redirect vector programming operations to these user vectors. The user's code should therefore continue to implement the normal (non-monitor) vector locations ($FF80-$FFFE). If execution of an interrupt with an un-programmed vector is attempted, behavior is undefined. For this reason, the user is strongly encouraged to implement a software trace for all vectors, as is good programming practice. The monitor depends on interrupts being available for monitor re-entry after GO or TRACE commands. Therefore, it is important that the user application executes with interrupts enabled.

APPENDIX B - UBUG12 COMMAND LIST

---------- **REGISTERS** ----------

```
RD                                          - Register Display
RM      <RegisterName> <Data8/16>           - Register Modify
CCR     <Data8>                             - Set CCR register
D  <Data16>                                 - Set D register
PC <Data16>                                 - Set PC register
PP <Data8>                                  - Set PP register
SP <Data16>                                 - Set SP register
X  <Data16>                                 - Set X register
Y  <Data16>                                 - Set Y register
```

---------- **MEMORY MODIFY** ----------

```
BF <StartAdd> <EndAdd> <Data8>              - Block fill
BFW     <StartAdd> <EndAdd> <Data16>        - Block fill word
MD <StartAdd> [<EndAdd>]                    - Memory display
MDW     <StartAdd> [<EndAdd>]               - Memory display word
MM <Address> <Data8>                        - Memory modify byte
MMW     <Address> <Data16>                  - Memory modify word
```

---------- **FLASH** ----------

```
FBULK                                       - Flash bulk erase
FLOAD   [;B][;M]                            - Flash load
```

---------- **DEVICE INFO** ----------

```
DEVICE                                      - Get device name
```

---------- **GO/HALT** ----------

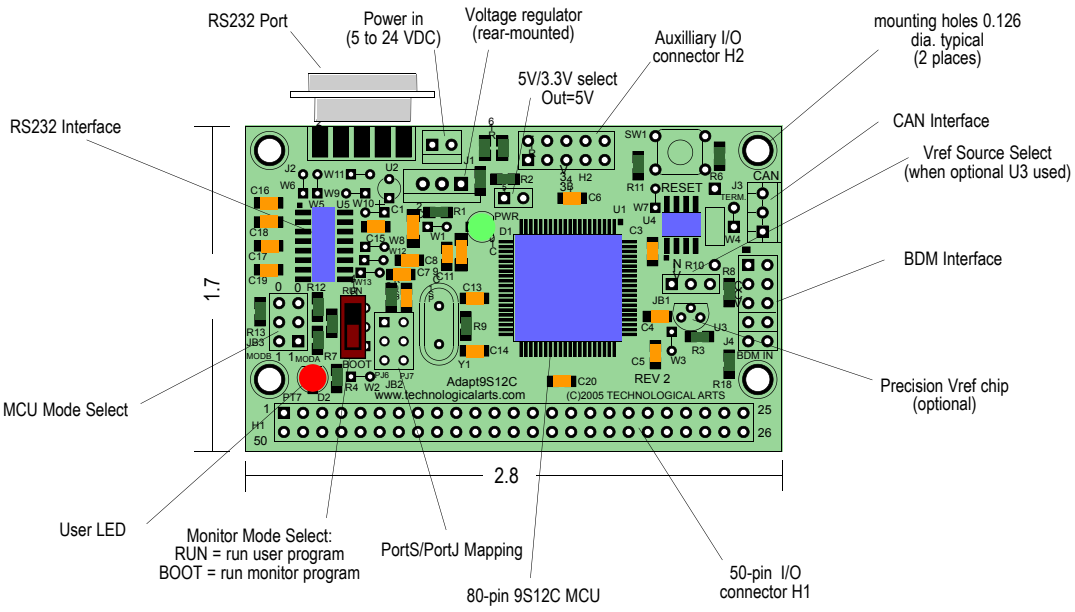
```
GO [<StartAddress>]                         - Start execution
HALT                                        - Halt execution
RESET                                       - Reset target
```

---------- **GUI** ----------

```
CON     <Comport>                           - Connect to target
DISCON                                      - Disconnect from target
EXIT                                        - Terminate GUI
HELP                                        - Display help
OP <Opacity%>                               - Set main GUI opacity
```

Adapt9S12C Module
Features and Pin Configuration

Dimensions in inches



H1 Pin Assignments

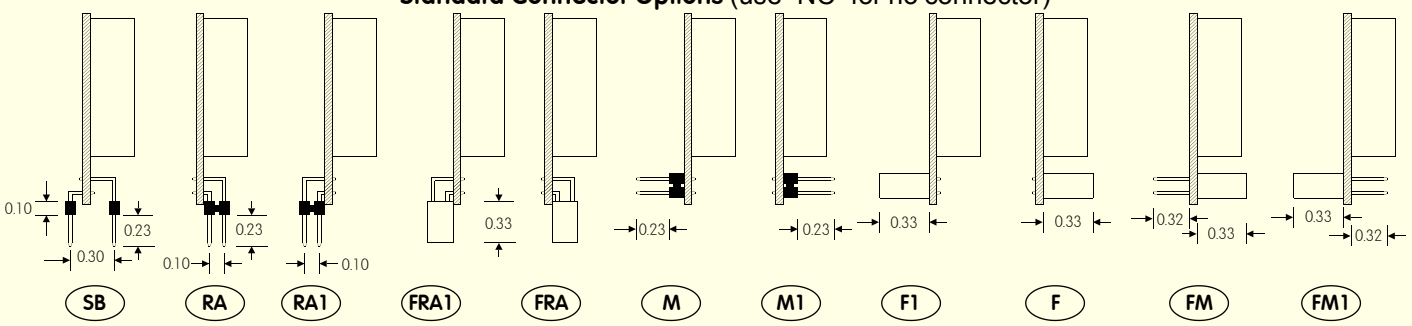
PIN #	NAME	PIN #	NAME
1	PM2/MISO	50	GROUND
2	PM4/MOSI	49	GROUND
3	PM5/SCK	48	PS0 or PJ6 (set by JB2)
4	PM3/SS*	47	Vcc
5	PS1 or PJ7 (set by JB2)	46	PE1 (IRQ*)
6	PT7/IOC7	45	PE0 (XIRQ*)
7	PT6/IOC6	44	RESET*
8	PT5/IOC5	43	PE7 (XECLKS*)
9	PT4/IOC4/PW4	42	PA0/ADDR8/DATA8
10	PT3/IOC3/PW3	41	PA1/ADDR9/DATA9
11	PT2/IOC2/PW2	40	PA2/ADDR10/DATA10
12	PT1/IOC1/PW1	39	PA3/ADDR11/DATA11
13	PT0/IOC0/PW0	38	PA4/ADDR12/DATA12
14	PB7/ADDR7/DATA7	37	PA5/ADDR13/DATA13
15	PB6/ADDR6/DATA6	36	PA6/ADDR14/DATA14
16	PB5/ADDR5/DATA5	35	PA7/ADDR15/DATA15
17	PB4/ADDR4/DATA4	34	PE2 (R/W*)
18	PB3/ADDR3/DATA3	33	PE4 (ECLK)
19	PB2/ADDR2/DATA2	32	PE3 (LSTRB*)
20	PB1/ADDR1/DATA1	31	Vana
21	PB0/ADDR0/DATA0	30	VRH
22	PAD00/AN00	29	PAD04/AN04
23	PAD01/AN01	28	PAD05/AN05
24	PAD02/AN02	27	PAD06/AN06
25	PAD03/AN03	26	PAD07/AN07

H2 Pin Assignments

PIN #	NAME	PIN #	NAME
1	PP0/KWP0/PW0	2	PP7/KWP7
3	PP1/KWP1/PW1	4	PP6/KWP6/ROMCTL
5	PP2/KWP2/PW2	6	PP5/KWP5/PW5
7	PP3/KWP3/PW3	8	PP4/KWP4/PW4
9	Vcc	10	GND

Order Codes:
AD9S12C32M-❑
AD9S12C128M-❑
(specify H1 connector option code)

Standard Connector Options (use "NC" for no connector)



www.technologicalarts.com • sales@technologicalarts.com • phone: +1 (416) 963-8996 • fax: +1 (416) 963-9179