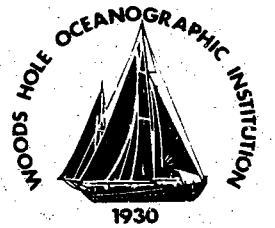
WHOI-90-33

Copy 2

Woods Hole Oceanographic Institution



The High Speed Buffer Board
A SAIL EIA-485 Communications Accelerator Card for the Vector Measuring Current Meter

by

Robin Singer and Douglas M. Butler

July 1990

Technical Report

Funding was provided by the Office of Naval Research through Contract No. N00014-84-C-0134, and Grant N00014-90-J-1495.

WHOI-90-33

The High Speed Buffer Board
A SAIL EIA-485 Communications Accelerator Card for the Vector Measuring Current Meter

by

Robin Singer and Douglas M. Butler

Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

July 1990

Technical Report

Funding was provided by the Office of Naval Research under Contract No. N00014-84-C-0134 and Grant No. N00014-90-J-1495.

 This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-90-33.

Approved for publication; distribution unlimited.

Approved for Distribution:

Albert J. Williams 3rd, Chairman
Department of Applied Ocean Physics and Engineering

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

ABSTRACT

A High Speed Buffer Board (HSBB) has been developed for the Vector Measuring Current Meter (VMCM) to implement the transmission of data at 9600 baud over an EIA-485 link. The HSBB significantly extends the VMCM communication functionality, which was previously limited to 300 baud transmission via 20 mA current loop or FSK telemetry. The increased speed allows rapid sampling of a large number of current meters on a common cable and the EIA-485 circuitry, which was designed for low power operation, provides a useful multipoint communication method for data transmission over long cable lengths. SAIL protocol (IEEE 997) was utilized to coordinate data transfer by the instruments on a common link.

An MC68HC11 microcontroller resides in the VMCM, buffering data it receives at 300 baud from the VMCM UART. In response to a jumper selectable SAIL address, the MC68HC11 offloads the data at 9600 baud via EIA-485 to the SAIL controller. Synchronous data collection from many instruments is ensured by the SAIL synoptic set command and an embedded resynchronization/reset command. The low power consumption allows deployments of six months or more with a standard VMCM battery stack.

I INTRODUCTION

1.1 Motivation

Physical oceanographers at the Woods Hole Oceanographic Institution and elsewhere have recently focused their attention on the dynamics of surface waves and wave related processes in the upper ocean. Experiments are underway to develop or refine models of near surface mixing, vertical transport, and velocity and density field variability. The influence of wind and wave conditions on such processes as Langmuir Circulation, near-surface shear, bubble clouds and turbulence, is being examined.

In order to establish these relationships, oceanographers needed data of greater temporal and spatial resolution than that of previous experiments. In particular, data collected by Vector Measuring Current Meters (VMCM's) had to be acquired at a higher rate, with closer spacing, over sufficient depth to resolve the structure of the near surface mixed layer. Therefore, a High Speed Buffer Board was developed for VMCM's as part of the Surface Wave Physical Processes Program (SWAPP), to provide transmission of velocity and temperature data at 9600 baud over an EIA-485 link using SAIL protocol. The board extends the use of VMCM's to the investigation of short time constant events.

1.2 Design Requirements

The HSBB was designed to enable up to 20 self-powered, uniquely addressed VMCM's to send a 39 character data string over 200 meters of sea cable, in under 2 seconds, for a period of up to two months. This requirement determined the features of the HSBB which include implementation of SAIL protocol and EIA-485 data telemetry, use of CMOS parts for low power consumption, and 9600 baud serial data communication capability. A further requirement was that the VMCM sampling interval could be periodically reset to prevent significant oscillator/time base drift over the course of a deployment. This led to the design of a resynchronization module which resets the VMCM when triggered by the HSBB upon receipt of a SAIL command over the sea cable.

II DESIGN DESCRIPTION

The HSBB was developed for use with the EG&G Model 630 Vector Measuring Current Meter equipped with the SAIL option. Figure 1 shows the major functional blocks. The schematic is included as Figure 3 on page 7.

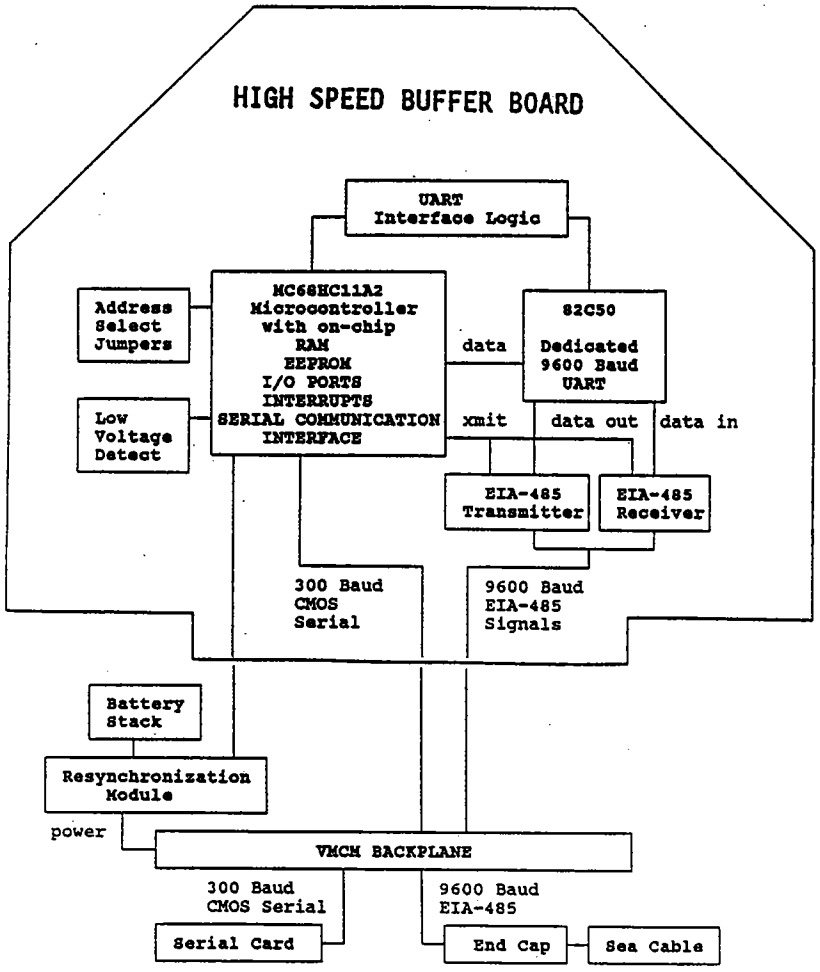


Figure 1.

2.1 Microcontroller and Related Circuitry

At the heart of the HSBB is a Motorola MC68HC11A2, a CMOS microcontroller with on-chip EEPROM, RAM, I/O ports, a serial communication interface, and interrupts. A 3.6864 MegaHertz crystal drives an internal clock generator circuit to provide a processor clock at one fourth the crystal frequency or 921.6 kiloHertz. Since power consumption is a function of clock speed, operating at this low frequency limits the power cost.

The microcontroller performs the major task of the HSBB, to receive 300 baud serial data from the VMCM, buffer it in RAM, and transmit it at 9600 baud in response to a SAIL request. The 68HC11's serial communication interface (SCI) is used for the 300 baud communication with the VMCM. Five bits in the Baud Rate Register determine pre-scaler and rate control values for generation of transmitter and receiver clocks from the processor clock. One of the five SCI interrupts, the Receiver Data Register Full interrupt, alerts the microcontroller when a character has arrived from the VMCM and should be stored in a RAM buffer.

The 68HC11A2 has 256 bytes of RAM, which can be relocated to any unused 4 kilobyte boundary in the microcontroller's memory space. In the HSBB, they are located at the default starting address (0000) where they provide stack space and storage space for variables and buffers. The HSBB memory map is shown below:

HSBB Memory Map

	Hexadecimal Addresses
RAM	0000-00FF
stack	0036-005E (grows down)
variables	005F-0070
buffers	009B-00FF
Registers	1000-103F
82C50 UART (memory mapped I/O)	9000-9FFF
EEPROM	F800-FFFF
code space	F800-F8BF
interrupt vectors	FFC0-FFFD
reset vector	FFFE-FFFF

Figure 2

The HSBB firmware is located in the 2 kilobytes of EEPROM which reside at hexadecimal adresses F800 to FFFF. The chip also has a built-in high voltage charge pump for EEPROM programming and erasure. The MC68HC11A2 was selected for use in the HSBB because the other members of the 68HC11 family had at most 512 bytes of EEPROM and would have required an additional EEPROM or EPROM chip to store the 750 bytes of code.

The HSBB has the 68HC11's MODA and MODB pins pulled high through 8.2 kilohm resistors which configures it for the expanded multiplexed mode of operation. This provides 64 kilobytes of address space using ports B and C as address lines. A 74HC373 (octal D latch chip) demultiplexes the low order address lines from the data lines which share Port C in this mode. The microcontroller's Address Strobe timing signal (AS pin) is used to enable the 74HC373 to latch the address lines when they appear on the Port C pins.

The address space generated in this mode is used for memory mapped I/O of the 82C50 dedicated UART, operated at 9600 baud. UART timing signals are generated from the microcontroller E clock and read/write pin (R/W), some CMOS logic gates, and an RC network. The UART is mapped to hexadecimal address 9xxx and the three lowest order address lines select the registers to write to or read from. A 3.6864 MegaHertz crystal provides the clock for the 82C50's baud rate generator. This frequency is the same as the microcontroller crystal so that if HSBB power consumption needs to be reduced, one of the crystals could be eliminated.

The SAIL address selection circuit makes use of eight input port pins and one output port pin. During initialization, the microcomputer raises port pin PA6 to energize one side of jumpers J4-J11. The other side of the jumper block is tied to ground. The middle posts, which can be jumpered to either side to indicate a "0" or a "1", are connected to port pins PD2-6 and PE0-3 and are read by the microcomputer to determine the SAIL address. When the address has been read, the microcontroller brings PA6 low again to conserve power.

Early tests of the MC68HC11 'A' series revealed a need for a low voltage inhibit circuit to prevent corruption of EEPROM contents in case of a low supply voltage (VDD). A SEIKO 8054 resets the 68HC11 if it detects a low power supply voltage thus preventing execution of instructions when VDD is too low to support proper operation. This circuit solves the problems potentially associated with the power on sequence and low battery conditions. Motorola has subsequently solved the EEPROM corruption problem and any future versions of the HSBB can leave out low voltage protection.

2.1.a Upgrade of Microcontroller

Motorola has discontinued production of the MC68HC11A2 and has replaced it with an upward compatible part, the XC68HC811E2. 48 pin DIPs

were used on the HSBB rather than the more compact 52 pin PLCC packaged chips, which were unavailable at the time the HSBB's were constructed. For future production runs, the 48 pin DIP version of the XC68HC811E2 can be substituted for the MC68HC11A2 without any changes.

2.2 EIA-485

2.2.a Choice of EIA-485

The requirements of remote oceanographic field work necessitate the development of low power telemetry systems for instruments. Two SAIL telemetry options were available for the VMCM prior to the design of the HSBB, 300 baud 20 milliamp current loop and 300 baud Frequency Shift Keying (FSK) telemetry. Neither of these options could easily be extended to 9600 baud over long cable lengths at low power so EIA-485 telemetry was selected for implementation.

EIA-485 is a two wire, multipoint, differential bus standard which allows data rates up to 10 Megabits per second over cable lengths up to 4000 feet and provides a high level of noise immunity. The modest requirements of the HSBB allowed the EIA-485 implementation to abandon the high data rate and cable length capabilities in order to achieve a low power consumption design which interfaces reliably with commercially available EIA-485 implementations.

2.2.b EIA-485 Circuitry

The interface between the EIA-485 and CMOS circuits on the HSBB consists of three lines: Data In, Data Out, and -XMIT. The Data signals are standard "+5" and "0" volt CMOS levels with a mark or a stop bit represented by a +3.5 - +5 volt line and a space or start bit represented by a 0 - 1 volt level. The - Xmit pin is active low, i.e. when a character is to be sent, the -Xmit pin must be pulled down to enable the line driver chips which buffer the Data signals.

Transmitter

When the -Xmit pin is low (active), the inverting buffers of the 74HC240 chip are enabled and both a buffered Data Out signal and an inverted buffered Data Out signal are supplied to the gates of transistors Q1, Q2, Q3, and Q4. Q1 - Q4 are two complimentary pairs of MOSFETS which drive EIA-485 terminals A and B oppositely. 120 Ohm resistors R2, R3, R6, and R7 limit the slew rate to reduce crosstalk in bundled cables, mask any line imbalance due to unequal Rdson of the FETs, and most importantly, limit the fault current in the case of a shorted line or bus contention. Using separate upper and lower resistors also reduces totem pole current spiking during transmitter transitions.

When the -Xmit line is high (passive), the 74HC240 is tristated, allowing the gate resistors to shut off Q1-Q4. Diodes D1-D4 prevent the substrate diodes of Q1-Q4 from conducting thereby isolating terminals A and B from the transmitting circuit. The use of gate resistors also assures that the transmitter will become passive upon loss of power.

Receiver

The receiver centers around an ICL 7611, a low power CMOS op amp. It is isolated from the EIA-485 lines by 1 Megohm resistors (R12 and R17) which protect it from damaging overloads. The EIA-485 standard requires that the receiver be able to function in the presence of +/- 7 volts of common mode noise. A balanced 8 to 1 voltage divider consisting of R12, R13, R15, R17, reduces the amplitude of the signal plus noise to prevent exceeding the common mode range of the op amp. R11, R14, and R16 set the bias points of the amplifier. R14 provides a slight offset to provide for a stop bit condition when no valid signal is being received, such as when all transmitters on the line are passive. C14, although not normally required, may aid in noise rejection in exceptionally high noise environments.

The trade-off between power consumption and slew rate of the amplifier is determined by pin 8 of the op amp, the quiescent current programming pin. Connecting this pin to the threshold voltage provides for a power consumption of only 100 microamperes while providing adequate slew rate for receiver operation at 9600 baud. The op amp output is compatible with CMOS logic chips and feeds directly into the 82C50 UART Serial Data Input line.

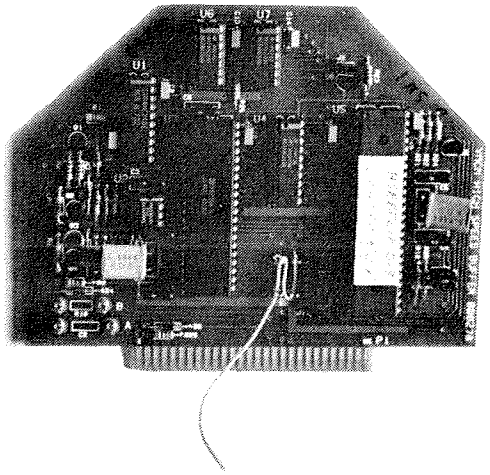
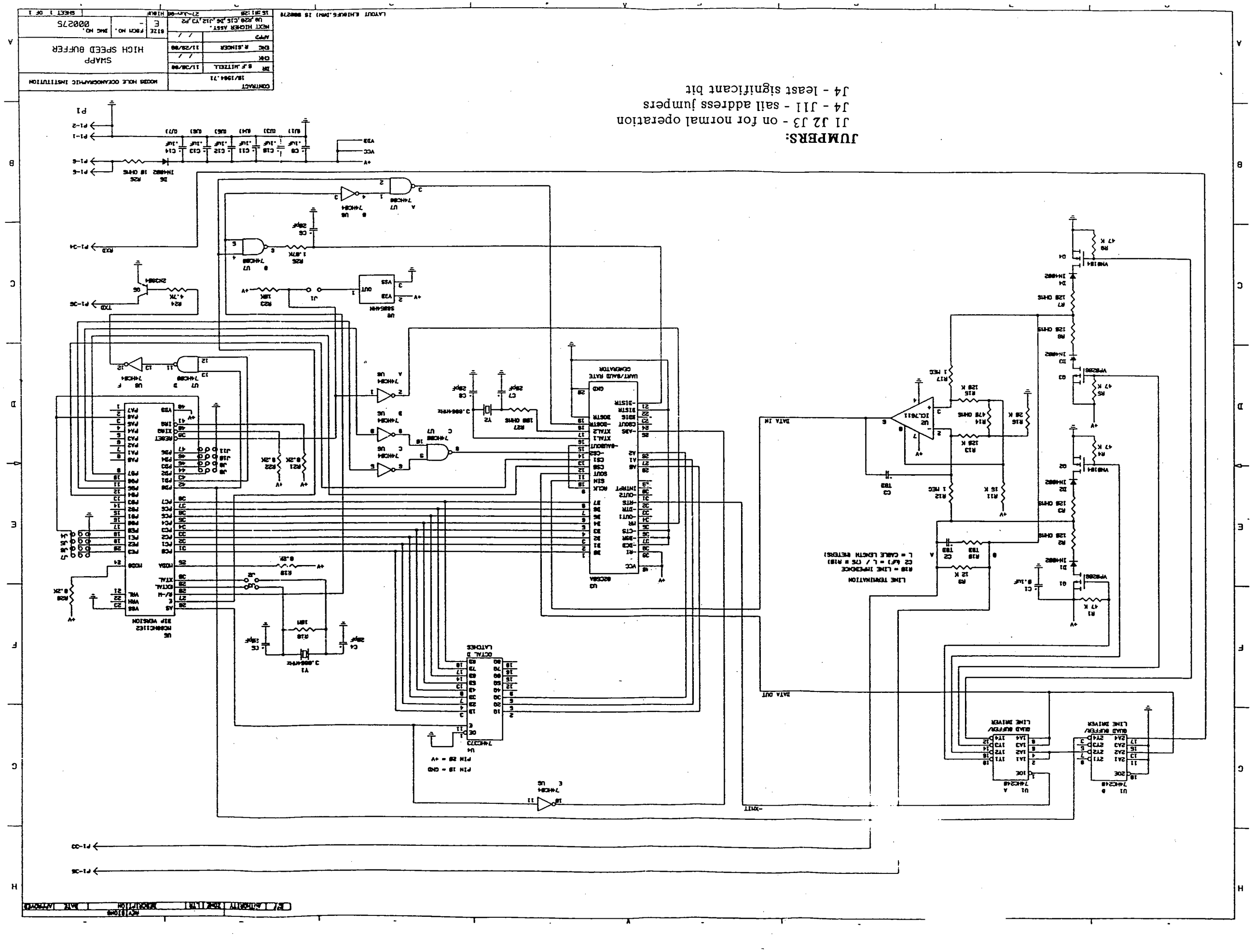


Figure 4. HSBB

FIGURE 3


JUMPERS:
J1 J2 J3 - on for normal operation
J4 - J11 - sall address jumpers
J4 - least significant bit
SHARP
HIGH SPEED BUFFER
HOOPS HOLE OCEANOGRAPHIC INSTITUTION
CONTRACT
18/1944.71
SHEET 1 OF 1
DATA IN
DATA OUT
LINE TERMINATION
LINE DRIVER
LINE RECEIVER
LATCHES
OCTAL D
PIN 20 = +V
PIN 10 = GND

2.3 Firmware

The firmware for the HSBB was written in the assembly language of the MC68HC11 on an IBM-PC compatible computer. It was assembled and converted to Motorola S19 format and downloaded to the 2 KiloBytes of EEPROM residing on the microcontroller I.C., using a Motorola Evaluation Module and the EVMbug monitor. The code takes up only 750 bytes so there is room for special purpose modification without adding additional code space. The microcontroller has 256 bytes of on-chip RAM of which 170 bytes were used for variables and stack space.

2.3.a SAIL Protocol

The firmware implements SAIL protocol (IEEE 997) which regulates data flow among multiple instruments on a common link. A central SAIL controller (typically a data acquisition computer) communicates with a given instrument by sending an Attention Character (#) followed by the instrument's unique two digit SAIL address. For the HSBB-configured VMCM, this address is set on HSBB jumpers and on a switch on the VMCM serial card. During initialization, the HSBB 68HC11 reads the jumper block which is tied to port pins PE0-PE3 and PD2-PD6.

The SAIL standard also includes a Synoptic Data Access function which allows the SAIL controller to cause all instruments on a link to simultaneously store data, to be acquired when the instruments are addressed individually. When the HSBB receives the Synoptic Set address, #79, it stores the location of the most recently acquired buffer of data in anticipation of a subsequent query from the SAIL controller. Upon receipt of an Attention character and its SAIL address followed by the VMCM data offload command (R), the HSBB transmits the stored VMCM data stream over the link. The SAIL address #20 has been set aside as a command to reset the HSBB-configured VMCM for resynchronization.

Table I summarizes the HSBB's SAIL implementation*:

Table I SAIL COMMANDS

Commands	Responses
#nmR	send data string
#79	"freeze" data buffer
#20	reset/resynchronize

(n and m are the ASCII characters of the SAIL address)

* When testing the HSBB with a terminal or computer, it is necessary to send a #79 before each #nmR.

2.3.b Triple Buffering

The VMCM is preset to convey data through its UART to the HSBB at a switch selectable interval. This data is triple buffered in the HSBB in order to ensure that no data will be lost as long as the SAIL controller sends the Synoptic Set command and a SAIL query over the link at that same preset interval. The triple buffering prevents data loss which could result from an offset between the start of the SAIL controller's interval and the start of the VMCM interval. A query must be sent within one sample interval after a Synoptic Set in order to ensure retrieval of the frozen data.

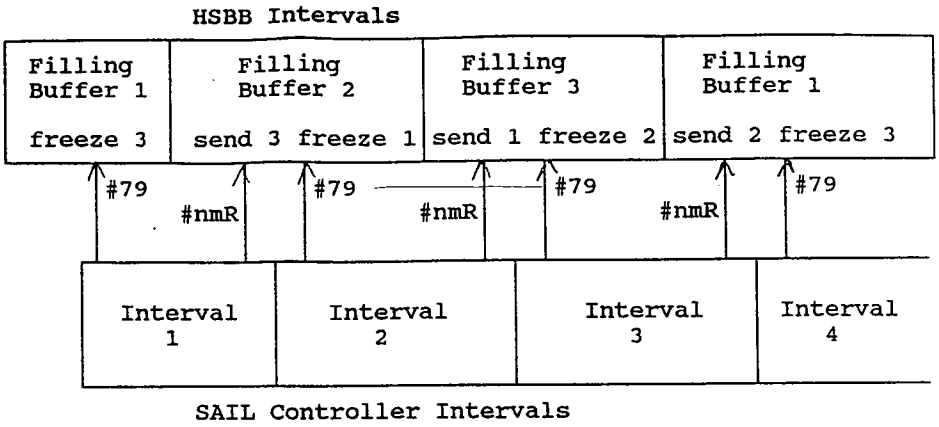


Figure 5.

During each HSBB interval, it is necessary to have one buffer which is being filled by incoming VMCM data, a second buffer which has been frozen and is ready to be sent upon receipt of #nmR, and a third buffer which has been filled and is ready to be frozen upon receipt of #79.

2.3.c Use of Interrupts

There are two asynchronous serial channels used simultaneously by the HSBB. The HSBB standalone UART, the 82C50, receives and transmits 9600 baud data over the EIA-485 SAIL link. The 68HC11 Serial Communications Interface (SCI) transmits a 300 baud command to start the VMCM data offload and then receives the 300 baud data sent by the VMCM at preset intervals.

The 68HC11 makes use of the SCI serial system interrupts to acquire the VMCM data. The receive data interrupt service routine stores data in the incoming data buffer and updates the buffer pointer when a new-line character is received. This frees up the microcontroller to continually poll the data ready bit in the line status register of the 82C50 and to send the appropriate data stream when a query is received.

2.3.d Firmware Modules

The HSBB firmware is listed in Appendix 1. It is described in detail below.

HSBBFLIP.ASM is a modular program made up of the following building blocks:

buffinit routine - initializes the input pointer by loading it with the address of the first of the three buffers.

sailinit routine - reads the SAIL address from the jumpers

ser_ init routine - sets up the 68HC11 Serial Communication Interface (300 baud) and the 82C50 Uart (9600 baud)

ctrdelay routine - half second delay routine for tweaking timing where necessary

prepvmcm routine - preparing VMCM for a command (guarding against a VMCM idiosyncrasy of missing the first characters sent after a reset)

startvmcm routine - instructing VMCM to offload data every 2 seconds through its serial port

addr routine - called by startvmcm, sends SAIL address to VMCM

sendchar routine - 68HC11 SCI transmit character routine

newbuf routine - puts proper buffer address into input buffer pointer

sendit routine - 82C50 UART transmit character routine

da routine - 82C50 UART receive character routine for polling of data available status

senbf routine - sends the most recently frozen buffer via the 82C50 UART

convbufno and convposn routines - for diagnostic/debug use

synch routine - sends pulse to resynchronization module to initiate a reset

freeze routine - stores a pointer to the most recently filled buffer after a SAIL synoptic set command was received

rcvint routine - interrupt service routine for the 68HC11 SCI interrupt, storing incoming characters and looking for the end of a data string.

2.3.e Firmware Synopsis

The program consists of three parts: initialization, SAIL listening and execution, and servicing of the SCI receive data interrupt. Initialization occurs after reset and consists of reading the SAIL address, initializations of the buffers and serial channels, and initiation of the VMCM automatic data offload.

In the main loop of the program, the microcontroller polls the 82C50 line status register to determine if a SAIL character has arrived. If so, it reads the character and determines if a SAIL command has been received. If a synoptic set command has been received, the freeze routine is called in order to store a pointer to the most recently filled buffer. If a data offload command has been received, the contents of the most recently frozen buffer are sent out the EIA-485 channel. If a resynchronization command has been received, the microcontroller calls the synch routine to reset the VMCM.

While the main loop is being executed, SCI receive interrupts can occur and the 300 baud VMCM data is collected and buffered. The last character in a data string will initiate a buffer pointer bookkeeping/reload by the newbuf routine. No characters should be missed on either serial channel because of the short duration of the program instruction cycles with respect to the transmission rate of the characters.

The source code listed in Appendix 1 also establishes the stack, variables, constants, and the interrupt vectors at the locations shown on the HSBB Memory Map (Figure 2).

2.4 Resynchronization Module

The HSBB can initiate a reset of the VMCM upon receipt of a SAIL command (#20). This feature enables the central SAIL controller to

resynchronize the data collection periods of the instruments, by issuing this command over the SAIL channel. When the HSBB receives the resynchronization command, the 68HC11 microcontroller raises port pin PA4 which is wired to a resynchronization module in line with the battery stack supplying power to the VMCM. The module interrupts the power to the VMCM, causing a hardware reset.

This somewhat radical method was employed due to unreliable responses of the VMCM SAIL diagnostic reset and the VMCM reset circuitry. The initial design made use of the SAIL diagnostic "G" command. In VMCM diagnostic mode, the command G0000 (go to location 0000) should reset the VMCM. However, this sequence did not always occur as documented and could not be relied upon for dependable resynchronization.

The second design made use of a 68HC11 port pin to pull down the CLR line of the VMCM's CDP1802 processor to initiate a reset. Although a hardware modification was made to pull down the CLR line according to the I.C.'s specifications, whenever a SAIL command was issued, the VMCM did not always respond with a reset. In order to achieve a reliable reset/resynchronization, the power down reset method was adopted.

The resynchronization module (shown in Figure 6) consists of a CD4098B CMOS Dual Monostable Multivibrator (one-shot), an IRF 9530 p-channel MOSFET, a pair of DB-9 connectors, and a one pin Molex connector which mates with a connector on a wire from pin PA4 of the 68HC11. The one shot is configured with a 2 microfarad capacitor and a 1.499 Megohm 1% resistor so that a trigger pulse issued by PA4 is converted into a 1.499 second pulse on the gate of the IRF9530. Since the source is at VDD (battery voltage), VGS is pulled from - VDD volts to 0 volts for the duration of the pulse. This turns the FET off and interrupts the power supply to the VMCM which is connected to the FET's drain. A female DB-9 connector on one side of the resynch module mates to the VMCM power connector and a male DB-9 connector on the other side of the module mates to a connector from the battery stack.

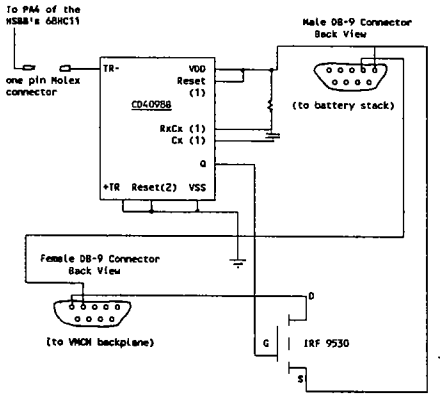


Figure 6. Resynchronization module

III SPECIFICATIONS

3.1 Power Consumption

The HSBB draws 10 milliamps at 5 volts (50 milliwatts). An HSBB-configured VMCM with the tape recorder and transport board removed draws between 14 and 19 milliamps at 5-6 volts and therefore consumes less than 114 milliwatts. Thus, a standard W.H.O.I. 168 Amp-hour VMCM battery stack would be discharged to about half of its capacity at the end of a six month deployment in an HSBB-configured VMCM.

3.2 Data Format

There are 39 ASCII characters in the string sent by the HSBB to the central SAIL controller.

The HSBB data string format is shown below:

#nmRF0rrrrnnnneeee2222llllccccttttCCcrlfetx

#nmR is an echo of the SAIL attention sign, address, and data offload command.

F0 is the preamble (always F0)

rrrr is the record number which ranges from 0002 to FFFF.
(Record 0001 is used for VMCM initialization.)

nnnn is the North Current Vector.

eeee is the East Current Vector.

2222 is the Rotor 2 count.

1111 is the Rotor 1 count.

cccc is the Compass reading.

tttt is the temperature reading.

CC is a longitudinal checksum.

CRLF is carriage return/line feed.

ETX is the ASCII end of text character used to indicate the end of a SAIL transmission.

The current version of the HSBB firmware appends six diagnostic characters at the end of the string, before the CR LF ETX characters. The comments at the top of the firmware listing shown in Appendix 1 detail these diagnostic characters. These characters provide for the determination of the resynchronization interval by indicating the rate of drift between the SAIL central controller time base and the VMCM clock.

IV USING THE HSBB

4.1 VMCM Backplane Modifications:

A 300 baud Frequency Shift Keying (FSK) telemetry card, previously developed for the VMCM, required several modifications to the backplane to convey FSK signals between the endcap and the transciever and to interface serial data between the telemetry card and the VMCM serial card. (see Reference 1) To maintain compatibility with FSK modified current meters, the HSBB utilizes the same modified backplane.

The modifications are shown in Figure 7. Wires are soldered between the backplane's J12 (endcap signal connector) and J8, the backplane connector for the HSBB. These transfer EIA-485 signals between the interconnecting EIA-485 cable and the HSBB's EIA-485 transciever. Jumper wires are also added between J8 and J7, the backplane connector for the serial card. These wires transfer CMOS level serial data signals between the HSBB and serial card. Corresponding traces between J8 and J9 must be cut since these signals should not be connected to the compass interface board.

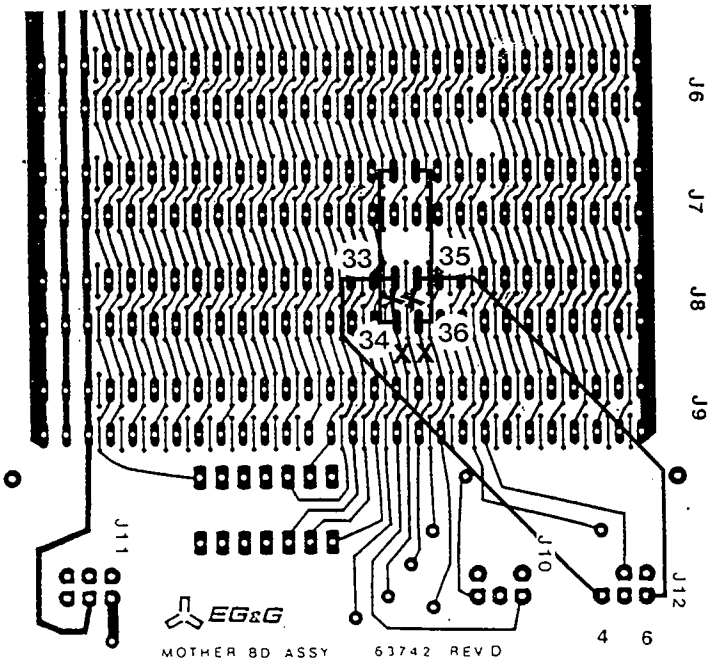


Figure 7. VMCM Backplane Modification

An "open collector" 2N3904 (npn bipolar junction transistor) on the HSBB drives the signals which go to the VMCM serial card. The collector is tied to pin 36 on J8 and the signal enters the serial card at pin 35 on J7 where a 100 kilohm resistor (R13) functions as a pullup resistor. (The 20 ma current loop circuitry, which also provides a signal at that point, is not altered and can still be used for communication when the EIA-485 channel is not in use). The signals are gated through a 4070 (CMOS exclusive-or gate) to the VMCM's CDP1854 UART Serial Data In (SDI) pin. CMOS serial signals from the 1854's Serial Data Out (SDO) pin are inverted by a 4049 (CMOS inverter) and travel through J7 pin 33 to J8 pin 34 to the HSBB.

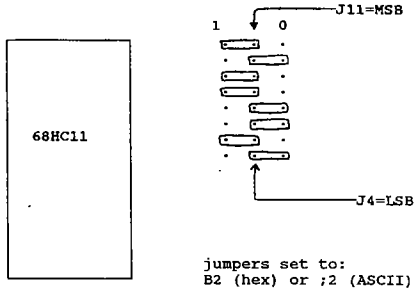
4.2 Switch Settings and Jumpers

4.2.a HSBB Jumper Settings:

There are 11 jumpers to install on the HSBB prior to operation.

Jumpers 1-3 are always installed for normal use. However, they must be removed when using a Motorola Evaluation Module (EVM) for debug purposes. J1 connects the low voltage detection circuit to the microcontroller RESET pin and must be an open circuit during EVM-assisted debug, to avoid contention with the EVM circuitry. J2 and J3 connect the microcontroller crystal to the EXTAL and XTAL pins and must also be removed when using the EVM, since an external oscillator is required to drive the EVM circuitry through the test cable length.

Jumpers 4-11 select the SAIL address, which must match the SAIL address on the 8 position switch on the VMCM serial card. (see section 4.2.b below) Each jumper can be configured as a "1" (connecting its corresponding port pin to PA6 which is brought to +5V during intialization) or configured to a "0" (connecting the port pin to ground). When the jumper connects the two pins closest to the 68HC11, it is configured as a "1" and when the two pins furthest from the 68HC11 are shorted, a "0" has been chosen. (see Figure 8)



HSBB SAIL ADDRESS JUMPERS

Figure 8.

Jumpers 4-7 represent the hexadecimal value of the lower order address digit and J8-J11 represent the hexadecimal value of the upper order address digit. To calculate the SAIL address from the jumpers, add 30 Hex (= 48 Decimal) to the hexadecimal values shown on the jumpers. Since each group of four jumpers can show values between 0 (jumpers set to 0000) and 15 (jumpers set to 1111), the addition of 30H results in hexadecimal numbers between 30 and 3F Hex (equivalent to 48-63 Decimal). Reference to an ASCII chart to translate the Hexadecimal (or Decimal) numbers to characters will yield two ASCII characters between "0" and "?". The characters can be any of the digits 0 through 9 or the characters listed below:

: ; < = > ?

To determine the jumper settings from the ASCII address characters, reverse the above procedure, i.e. look up the ASCII values on an ASCII table, subtract 30 Hex or 48 Decimal and set the jumpers to correspond to the resultant hexadecimal value. J11 is the most significant bit of the upper order address digit and J7 is the most significant bit of the lower order address digit.

```
example 1

jumpers: J11 J10 J9 J8     J7 J6 J5 J4
settings:  1   0   0  0      0  1  1  1
hex values:    8              7
ASCII values: 38 H (56 D)  37 H (55 D)
ASCII chars:   8              7
SAIL command for data offload = #87R

example 2

jumpers: J11 J10 J9 J8     J7 J6 J5 J4
settings:  1   0   1  1      0  0  1  0
hex values:    B              2
ASCII values: 3B H (59 D)  32 H (50 D)
ASCII chars:   ;              2
SAIL command for data offload = #;2R
```

Figure 9. Sail Address Examples

4.2.b VMCM Switches:

To operate the HSBB-configured VMCM, three eight-position DIP switches on the compass board and one eight position DIP switch on the serial board must be set. The eight-position DIP switch on the serial card sets the SAIL address, and must conform to the HSBB SAIL address jumper settings. Positions 1-4 on the switch are the low order SAIL digit (position 1 = least significant bit) and Positions 5-8 are the high order digit (position 8 = most significant bit). The address is derived from the switch setting as described above in section 4.2.a regarding HSBB jumpers 4-11. A "1" is set by the switch being pushed down on the "on" side with the red dot pushed up on the "off" side. A "0" is the switch pushed down on the "off" side. Switches 1-3 on the compass interface board set a variety of VMCM parameters including the record interval. To ensure the collection of all available data, this setting must correspond to the data acquisition interval used by the SAIL controller. Switch Bank 1 determines the record interval and test interval as described in the EG&G VMCM Model 630 Technical Manual (Reference 2). In particular, switches 1-4 of Switch Bank 1 determine the record interval which can be set to any one of sixteen prescribed periods ranging from 2 seconds to 2 hours. A 2 second interval is set by putting switches 1-4 all in the OFF position (switch pushed down on the OFF side) and a two hour interval is set when all 4 switches are in the ON position. The HSBB power-on sequence sends a 300 baud SAIL command, #nmR, (where n and m are the two ASCII SAIL address digits) which instructs the VMCM to automatically offload data through the serial port at the end of each record interval. The HSBB will acquire the data and buffer it for transmission upon receipt of a query from the SAIL controller over the 9600 baud EIA-485 channel. The test interval, which is used with the VMCM SAIL-diagnostic option, is not relevant to the functioning of the HSBB.

Switch Bank 2 contains the sample interval and the record options which determine the form of the VMCM data stream. The sample interval ranges from quarter second steps to 2 second steps and determines the frequency with which the compass is read and vector flow rates are calculated. The record options determine the composition of the data stream indicating whether pressure, temperature, auxiliary voltage, rotor counts, and compass values are reported along with the NORTH and EAST vector values. Switch Bank 3 is not used, with the exception of position 8 which identifies the presence or absence of the A/D option.

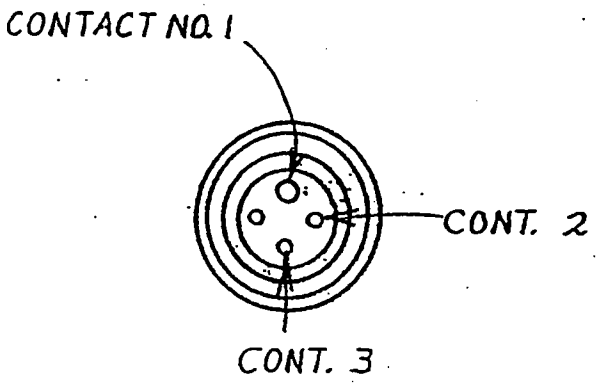
Note: Changing the data stream requires modification of the HSBB firmware, however, the record interval can be changed without any HSBB modifications.

4.3 Cables

The VMCM's configured at W.H.O.I. with High Speed Buffer Boards have a water tight Brantner Connector in the end cap (special order Dual SAIL Penetrator #6620-001). Four sockets are provided for four wire

communication (see figure 10). Contacts 2 and 4 are used for 20 milliamp current loop SAIL and contacts 1 and 3 are used for EIA-485 telemetry (or FSK communication in FSK-configured VMCM's).

The mating connector is Brantner number VMG-4-FS which can be ordered in an ether base polyurethane and fitted with a G-FLS-S locking sleeve. Pin 1 is 14 gauge and pins 2, 3, and 4 are 16 gauge. Pin 1 has been used as -D and pin 3 as D for the EIA-485 lines.



DATE	WOODS HOLE OCEANOGRAPHIC INSTITUTION		
DRAFT C. Marquette	BUOY GROUP ENGINEERING		
CHECKED	VMCM SAIL CABLE		
APPVD	EIA 485/FSK/CURRENT LOOP		
SCALE	DWG. NO. 10197.299	REV.	SHEET OF

Figure 10. VMCM End Cap Connector

4.4 Programming the EEPROM

The HSBB firmware is stored in the MC68HC11's 2 kilobytes of EEPROM, which resides at memory locations F800-FFFF Hex. The code was written with a text editor, assembled and linked on an IBM compatible computer with the 68C11 Assembler by 2000 A.D. Software, and downloaded to the microcontroller in Motorola S19 format. The linker was instructed to locate the code at address F800.

To download code to the HSBB, it is necessary to have a P.C., a terminal emulator program, a serial port to DB25 socket cable, and a Motorola Evaluation Module (EVM) which comes with the EVMbug Monitor and programming sockets. The 68HC11's are socketed on the HSBB for ease of programming and replacement.

The MC68HC11A2 must be inserted in the DIP programming socket on the EVM for erasure, programming and verification. (Early versions of the EVM had a trace missing to the ground pin on the DIP programming socket. This should be checked if any difficulty arises.)

The instructions for code assembly and linking and EEPROM erasure, programming, and verification are shown in Appendix 5. They give the commands for use with the Kermit terminal emulator which was distributed by Motorola to EVM purchasers.

4.5 Testing the HSBB-configured VMCM

The following equipment is useful for testing the HSBB-configured VMCM:

VMCM EIA-485 endcap connector
EIA-485 to EIA-232 conversion box
computer with EIA-232 (COM) port
EIA-232 breakout box
VMCM card extender
Test software

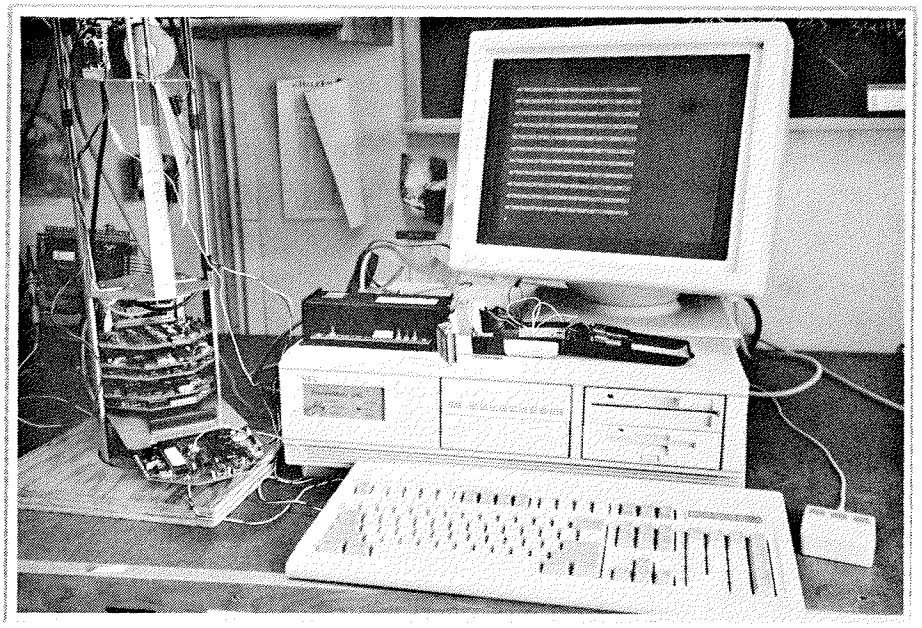


Figure 11.

The breakout box can be used to provide an appropriate interface between the computer and 485/232 conversion box.* Test software for an IBM compatible computer, developed under Microsoft Quick Basic Version 4.5 is listed in Appendix 2.

For the NEC Powermate 386 and Acromag Converter Model 4SCC-TMM-1, the connection is as follows: cross pins 2 and 3, cross pins 4 and 5, and connect pin 8 from the conversion box side to pins 6 and 20 on the computer side.

4.6 Revising the HSBB Firmware

The Motorola EVM can be used for emulation while making revisions to the HSBB firmware. HSBB jumpers 1, 2, and 3 can be removed to allow the use of the EVM with an external oscillator.

V SUGGESTIONS FOR FUTURE REVISIONS

Any future production runs of the HSBB can incorporate the following changes:

1. The XC68HC811E2 microcontroller should be used and the low voltage detection circuitry can be eliminated.

2. The artwork can be revised to label the MSB and LSB on the address jumpers (J4-J11) and also to label the position of the "0" and "1" connections.

3. The design can be changed to eliminate one of the two crystals which would reduce power consumption.

References

1. Fucile, Paul and Valdes, James, "An FSK Telemetry-Module for Vector Measuring Current Meters", Technical Report WHOI-87-55.

2. EG&G Environmental Equipment Division, Sealink Systems, Vector Measuring Current Meter Model 630 Reference Manual", May 1982.

3. EG&G Environmental Equipment Division, Sealink Systems, "Vector Measuring Current Meter SAIL/Diagnostic Manual", May 1982.

4. Electronic Industries Association, "Standard for Electrical Characteristics of Generators and Receivers For Use In Balanced Multipoint Systems (EIA-485)", April 1983.

5. Weller, Robert, Research Proposal,"The Role of Surface Waves in Determining Near-Surface Structure", June 1987.

Acknowledgments

We would like to thank Bob Weller, Al Plueddemann, Melora Park, Paul Fucile, Ed Mellinger and Steve Liberatore for ideas and assistance in the development of the HSBB.

```
;hsbbflip.asm
;by Robin C. Singer
;July 3, 1989

;This is the version for the test cruise - sending 8 extra
;characters for diagnostic purposes.  They will be sent
;before the CR LF ETX when the hsbb receives #nmR.

;This version resynchs again if it receives two #79's
;before it has received any characters from the vmcm
;after a resynch.

;The eight extra characters will be as follows:
;       1) freeze pointer when #79 received
;       2) freeze pointer when #nmR received
;       3,4,5) posn of inptr (ptr for incoming vmcm data) when #79 rcv'd
;       6,7,8) posn of inptr when #nmR received
;       -inptr posn is: buffer no. followed by 2 dig. hex no.
;        showing the position in the buffer

;The commands that the central SAIL computer can send to the
;hsbb are as follows:
;       #nmR    to get the vmcm whose addr is nm to send data
;       #79     to tell the vmcms to freeze i.e. store a pointer
;               to the most recently filled buffer
;       #20     to tell the vmcms to reset (for synchronization)

;uart register addresses

LCR     EQU     $9003   ;line control reg.
LSR     EQU     $9005   ;line status reg.
MCR     EQU     $9004   ;modem control reg.
DLL     EQU     $9000   ;baud rate divisor latch l.s. byte (when LCR(7)=1)
DLM     EQU     $9001   ; "   "     "      "  m.s. byte (when LCR(7)=1)
RBR     EQU     $9000   ;receiver buffer register (when LCR(7)=0)
THR     EQU     $9000   ;transmitter holding register (when LCR(7)=0)

;hc11 sci register addresses

RCVDAT  EQU     $102F
TXDAT   EQU     $102F
SCON1   EQU     $102C
SCON2   EQU     $102D
SSTAT   EQU     $102E
BAUD    EQU     $102B

;ASCII constants

ATTN    EQU     $23
ETX     EQU     $03

;memory allocation constants

STRLN   EQU     33      ;vmcm string length and size of data buffers
DATALN  EQU     30      ;length of string without cr lf etx
BUFFLN  EQU     1+33+33+33              ;3 buffers + extra byte
B_ORG   EQU     $FF-BUFFLN+$800         ;buffer origin at highest RAM loc
                                        ;will be added to F800 by locator
```

```
;port addresses for digital i/o

PORTA   EQU     $1000   ;port A data register
PORTD   EQU     $1008   ;port D data register
PORTE   EQU     $100A   ;port E data register

;timer/counter register addresses

TMASK2  EQU     $1024
TFLAG2  EQU     $1025

;set up stack in RAM between $36 and $64

        ORG     $836            ;added to F800 this yields $36
        RMB     40
stack:  RMB     1               ;stack grows down from $5E

;reserve space for variables

inptr:  RMB     2               ;pointer to buffer for receiving
outptr: RMB     2               ;pointer to buffer for sending out
iptrno: RMB     1               ;buffer number for in pointer
optrno: RMB     1               ;buffer number for out pointer
digl:   RMB     1               ;first digit of SAIL address
dig2:   RMB     1               ;second digit of SAIL address
frz79:  RMB     1               ;freeze buffer pointer when #79 sent
frznm:  RMB     1               ;freeze buffer pointer when #81 sent
in79:   RMB     1               ;inptr buffer no. when #79 sent (ascii hex)
inp79:  RMB     2               ;inptr buffer posn when #79 sent
innm:   RMB     1               ;inptr buffer no. when #nmR sent
inpnm:  RMB     2               ;inptr buffer posn when #nmR sent
flag:   RMB     1               ;flag to indicate that vmcm has responded
flag2:  RMB     1               ;flag to indicate that this is 2nd #79

;set up buffers at highest on-board RAM locations i.e. $9B to $FF

        ORG     B_ORG           ;added to F800 this yields 9B
buff0:  RMB     STRLN
buff1:  RMB     STRLN
buff2:  RMB     STRLN
endbuf: RMB     1               ;last location in RAM i.e. $FF


        ORG     $00             ;start code at F800 using linker

;initialization

        SEI                     ;disable interrupts
        LDS     #stack          ;set stack pointer
        JSR     buffinit        ;initialize buffer pointers
        JSR     sailinit        ;find out SAIL address from jumpers
        LDAA    #$00            ;indicate that we have just started
        STAA    flag            ; or restarted
        STAA    flag2           ;and no #79 without vmcm char in
        JSR     ser_init        ;initialize serial ports (sci and uart)
        JSR     ctrdelay        ;wait about half a second
        JSR     prepvmcm        ;prepare the vmcm
        JSR     startvmcm       ;tell it to send data every 2 sec.
        CLI                     ;enable interrupts
```

```
;take in characters from 82C50 SIN line till a # sign is available
;when # is received find out what command has been sent
;and respond accordingly

chkda:  JSR     da      ;check if data available i.e. char in A
        CMPA    #$00
        BEQ     chkda
        CMPA    #ATTN   ;is it a #
        BNE     chkda
got#:   JSR     da      ;received a #, get next char
        CMPA    #$00
        BEQ     got#
        CMPA    #ATTN   ;is it another #
        BEQ     got#
        CMPA    #$37    ;is it a 7
        BEQ     got7
        CMPA #$32  ;is it a 2
        BEQ got2
        CMPA    digl    ;is it digit 1 of the SAIL address
        BNE     chkda
gotdl:  JSR     da
        CMPA    #$00
        BEQ     gotdl
        CMPA    #ATTN   ;is it a #
        BEQ     got#
        CMPA    dig2    ;is it digit 2 of the SAIL address
        BNE     chkda
gotd2:  JSR     da
        CMPA    #$00
        BEQ     gotd2
        CMPA    #ATTN   ;is it a #
        BEQ     got#
        CMPA    #$52    ;is it an R
        BNE     not_r
        JSR     sendbf
not_r:  BRA     chkda
bg:     BRA     got#
got7:   JSR     da
        CMPA    #$00
        BEQ     got7
        CMPA    #ATTN
        BEQ     got#
        CMPA    #$39    ;is it a 9
        BNE     not9
        JSR     freeze  ;store which buffer to send at next #nmR
        LDAA    flag
        CMPA    #$01
        BEQ     bc
        LDAA    flag2
        CMPA    #$02    ;is it the 2nd #79 without a vmcm response?
        BNE     put2
        JSR     synch
put2:   LDAA    #$02
        STAA    flag2
        BRA     bc
not9:   CMPA    dig2  ;if digl was a 7 this might be a SAIL addr.
        BNE     bc
        LDAA    digl
```

```
        CMPA    #$37    ;is digl a 7?
        BEQ     gotd2   ;if so we got the SAIL address
bc:     BRA     not_r   ;same as BRA to chkda
got2:   JSR     da
        CMPA    #$00
        BEQ     got2
        CMPA    #ATTN
        BEQ     bg
        CMPA    #$30    ;is it a 0
        BNE     not0
        JSR     synch   ;resynchronize by resetting the vmcm
not0:   CMPA    dig2    ;if digl was a 2 this might be a SAIL addr.
        BNE     bc
        LDAA    digl
        CMPA    #$32    ;is digl a 2?
        BEQ     gotd2   ;if so we got the SAIL address
        BRA     bc

;buffer pointer initialization routine

buffinit:       LDX     #buff0
                STX     inptr   ;first point to buffer 0
                LDAA    #00
                STAA    iptrno  ;in pointer is starting at 0
                RTS

;routine to read SAIL address from jumpers

sailinit:       LDX     #PORTA
                BSET    0,X,$40         ;set PA6
                LDAA    #$0F
                LDX     #PORTE
                ANDA    X               ;now A has dig2
                ADDA    #$30            ;turn it to ASCII
                STAA    dig2
                LDAB    #$3C
                LDX     #PORTD
                ANDB    X
                LSRB
                LSRB                    ;now B has digl
                ADDB    #$30            ;turn it to ASCII
                STAB    digl
                LDX     #PORTA
                BCLR    0,X,$40         ;clear PA6


;serial port initialization routine - 68hc11 sci and 82C50 uart

;initialize the uart to 9600 baud, no parity, one stop bit, 8 data bits
;initialize the sci to 300 baud, no parity, one stop bit, 8 data bits

ser_init:
;initialize hc11 sci
        ;       LDAA    #$35    ;set 300 baud - for EVM 8M crystal
                LDAA    #$16    ;set 300 baud - for targ 3.684 crystal
                STAA    BAUD
                LDAA    #$00
                STAA    SCON1
                LDAA    #$2C    ;enable receive interrupt
                STAA    SCON2
```

```
;initialize uart
                LDAA    #$83    ;chooses parity,stop,data bits and sets DLAB
                STAA    LCR
                LDAA    #$18    ;for 9600 baud
                STAA    DLL
                LDAA    #$00    ; "
                STAA    DLM
                LDAA    #$03    ;bring DLAB back to 0 for access to RBR and THR
                STAA    LCR
                RTS

;routine to delay about 1/2 second

ctrdelay:       LDX     #0
not_over:       LDAA    TFLAG2
                ANDA    $80
                BEQ     not_over
                STAA    TFLAG2
                INX
                CPX     #51
                BLT     not_over
                RTS

;routine to send characters to the vmcm to prepare it for its address

prepvmcm:       LDAA    #ATTN
                JSR     sendchar
                JSR     sendchar
                RTS

;routine to send #nmR to the VMCM to start it going

startvmcm:      JSR     addr
                LDAA    #$52
                JSR     sendchar
                RTS

;routine to send vmcm a # and its SAIL addr

addr:           LDAA    #ATTN
                JSR     sendchar
                LDAA    dig1
                JSR     sendchar
                LDAA    dig2
                JSR     sendchar
                RTS


;routine to send out the character in the accumulator through
; the 68HC11 serial port (sci).


sendchar:       LDAB    SSTAT   ;read status
                BITB    #$80
                BEQ     sendchar
                STAA    TXDAT
cktc:           LDAB    SSTAT
                BITB    #$40
                BEQ     cktc
                RTS
```

```
;buffer bookeeping routine

newbuf:         PSHB
                PSHX
                PSHY
                LDAB    iptrno  ;in pointer number into B
                INCB
                CMPB    #$03
                BNE     ok
                LDAB    #$00    ;buffers are 0,1,2 - if 3 wrap to 0.
ok:             STAB    iptrno
                LDX     #btable         ;address of buffer table
                ABX
                ABX                     ;add 2*iptrno (2 bytes in addr)
                LDY     X
                STY     inptr
                PULY
                PULX
                PULB
                RTS


;routine to send out the character in A via 82C50 SOUT

sendit: PSHA
        PSHB
loop:   LDAB    LSR
        BITB    #$20    ;TRH empty?
        BEQ     loop
        LDAB    #$02    ;lower rts (-xmit) for use by 485 circuit
        STAB    MCR
        STAA    THR
chkemp: LDAB    LSR
        BITB    #$40    ;TEMT true? i.e. transmitter empty?
        BEQ     chkemp
        LDAB    #00     ;bring rts (-xmit) back up
        STAB    MCR
        PULB
        PULA
        RTS

;routine to see if a char is available by checking the data
; ready bit in the 82C50 line status register.  If no char
; is available a 00 gets sent back in accumulator A.  If
; a char has been received it is sent back in A.

da:     LDAA    LSR
        ANDA    #$01
        BEQ     ret
        LDAA    RBR
ret:    RTS


;routine to send out the last previously frozen buffer
;via the 82C50 uart.

sendbf: LDD     outptr
        JSR     convbufno
        STAA    frznm
```

```
        LDD     inptr
        JSR     convbufno
        STAA    innm
        LDD     inptr
        JSR     convposn
        STD     inpnm
        LDAA    #ATTN           ;first echo #nmR
        JSR     sendit
        LDAA    dig1
        JSR     sendit
        LDAA    dig2
        JSR     sendit
        LDAA    #$52
        JSR     sendit
        LDAB    #DATALN         ;vmcm data string length
        LDX     outptr
tloop:  LDAA    X
        JSR     sendit
        INX
        DECB
        BNE     tloop
        LDAA    frz79
        JSR     sendit
        LDAA    frznm
        JSR     sendit
        LDAA    in79
        JSR     sendit
        LDD     inp79
        JSR     sendit
        TBA
        JSR     sendit
        LDAA    innm
        JSR     sendit
        LDD     inpnm
        JSR     sendit
        TBA
        JSR     sendit
        LDAA    #$0D
        JSR     sendit
        LDAA    #$0A
        JSR     sendit
        LDAA    #ETX
        JSR     sendit
        RTS


;find out the buffer no. from the pointer in D
;and put it in A

convbufno: CPD #buff1
        BLT bzero
        CPD #buff2
        BLT b_one
        LDAA #$32       ;it is buffer 2 so send a 2
        RTS
b_one:  LDAA #$31       ;it is buffer 1 so send a 1
        RTS
bzero:  LDAA #$30       ;it is buffer 0 so send a 0
        RTS
```

```
convposn: CPD #buff1
          BLT bz
          CPD #buff2
          BLT b_
          SUBD #$DD     ;make it a number between 0 and 31
          BRA allb
b_:       SUBD #$BC     ;make it a number between 0 and 31
          BRA allb
bz:       SUBD #$9B     ;make it a number between 0 and 31
allb:     TBA           ;actually fit in A so copy it into B too
          ANDB #$0F
          CMPB #$09     ;is it A-F?
          BLE  less
          ADDB #$37     ;make it 'A' through 'F'
          BRA  lsd
less:     ADDB #$30     ;least signif digit
lsd:      ANDA #$F0
          CLC
          RORA
          RORA
          RORA
          RORA
          CMPA #$09
          BLE  lte
          ADDA #$37
          BRA  rtn
lte:      ADDA  #$30     ;most signif digit
rtn:      RTS


;routine to synchronize the vmcm's after receiving #78

synch:          LDX  #PORTA
                BSET 0,X,$10
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                NOP
                BCLR 0,X,$10
                JSR     ctrdelay        ;power on reset will occur and
                RTS                     ; the program will start over

;the freeze routine stores in outptr, the buffer pointer to
; the most recently filled buffer, for sending data to the
; SAIL controller on next receipt of #nmR.

freeze: LDAB    iptrno
        DECB
        BPL     not00
        LDAB    #$02    ;if buffer 0 wrap around down to buffer 2
```

```
not00:  LDX     #btable
        ABX
        ABX             ;adding 2*(iptrno-1) since 2 bytes in addr
        LDY     X
        STY     outptr  ;this buffer ptr used for synoptic freeze
        LDD outptr
        JSR convbufno     ;puts ascii for buffer no. in A
        STAA frz79
        LDD inptr
        JSR convbufno
        STAA in79
        LDD     inptr
        JSR  convposn  ;puts ascii for posn in D - msb in B
        STD inp79
        RTS

;this is the interrupt service routine for the sci receive int

rcvint: PSHA
        PSHX
        LDAA    SSTAT
        LDAA    RCVDAT
        LDX     inptr
        STA     X               ;use inptr as pointer and store char
        INX
        STX     inptr
        CMPA    #$0A            ;is it line feed? (should be 32nd char)
        BNE     return          ;branch if not lf
        LDAA    #ETX            ;end of text as per SAIL protocol
        STA     X
        JSR     newbuf          ;otherwise go to next buffer
        LDAA    #$01
        STAA    flag
return: PULX
        PULA
        RTI


btable: FDB     buff0
        FDB     buff1
        FDB     buff2

        ORG     $07D6
RCVV:   FDB     rcvint
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda
```

```
        FDB     chkda
        FDB     chkda
        FDB     chkda
        FDB     chkda

        ORG     $07FE           ;becomes FFFE since linked at F800
RSET:   FDB     $F800
```

```
DECLARE FUNCTION rec% ()
DECLARE FUNCTION aok% (strng$)

' Testhsbb.bas
' December 29, 1989
' Robin C. Singer
' Test program for hsbb's.  (Assumes a 2 second data period).


DEFINT A-Z

CONST TRUE = -1
CONST FALSE = 0
CONST CLR$ = "


COMMON SHARED prevrec$, thisrec$, starting

freez$ = "#79"
synch$ = "#20"
respons$ = ""
starting = TRUE
rrsp = TRUE


CLS
LOCATE 8, 17
PRINT "Testing the High Speed Buffer Board"

sail$ = " "
WHILE (LEN(sail$) <> 2)
  LOCATE 11, 1
  INPUT "Enter SAIL address of HSBB (2 ASCII digits):  ", sail$
  IF (LEN(sail$) <> 2) THEN
    LOCATE 11, 1
    PRINT CLR$
    END IF
WEND
addr$ = "#" + sail$ + "R"

comport$ = "0"
WHILE (comport$ <> "1") AND (comport$ <> "2")
   LOCATE 12, 1
   INPUT "Which COM port? enter 1 or 2:  ", comport$
   IF (comport$ <> "1") AND (comport$ <> "2") THEN
      LOCATE 12, 1
      PRINT CLR$
   END IF
WEND

count = 0
WHILE (count = 0) OR (count < -1)
  LOCATE 13, 1
  INPUT "How many queries before resynch? (pos. int. or -1 for no resynchs):  ", quer
  count = VAL(queries$)
  IF count = 0 THEN
    LOCATE 13, 1
    PRINT CLR$
  END IF
```

```
WEND

brc$ = " "
WHILE brc$ <> "y" AND brc$ <> "Y" AND brc$ <> "N" AND brc$ <> "n"
  LOCATE 14, 1
  INPUT "Turn on bad record counting? (y/n):  ", brc$
  IF brc$ <> "y" AND brc$ <> "Y" AND brc$ <> "N" AND brc$ <> "n" THEN
     LOCATE 14, 1
     PRINT CLR$
  END IF
WEND

CLS
IF brc$ = "Y" OR brc$ = "y" THEN
    LOCATE 12, 18
    PRINT "Press ESC to get bad count record"
END IF
LOCATE 14, 22
PRINT "Press ESC to exit"
LOCATE 16, 20
PRINT ""

' Open communications (9600 baud, no parity, 8-bit data,
' 1 stop bit, 256-byte input buffer)

IF comport$ = "1" THEN
  OPEN "COM1:9600,N,8,1" FOR RANDOM AS #1 LEN = 256
END IF

IF comport$ = "2" THEN
  OPEN "COM2:9600,N,8,1" FOR RANDOM AS #1 LEN = 256
END IF


badcount = 0
counter = 0
tick = FALSE
ON TIMER(1) GOSUB ticktock
TIMER ON

DO

   KeyInput$ = INKEY$            ' Check the keyboard.

  IF KeyInput$ = CHR$(27) THEN     ' Exit the loop if the user
     EXIT DO                        ' pressed ESC

  ELSE


      IF tick = TRUE THEN
         PRINT #1, freez$;
         PRINT freez$;
         tick = FALSE
         IF rrsp = FALSE THEN       'haven't yet received response
            badcount = badcount + 1
         ELSE
            rrsp = FALSE
         END IF
         IF counter = count THEN
```

```
            counter = 0
            PRINT synch$;
            PRINT #1, synch$;
         ELSE
            counter = counter + 1
         END IF


         WHILE tick <> TRUE
         WEND
         PRINT #1, addr$;
         PRINT addr$;
         tick = FALSE

      END IF


   IF NOT EOF(1) THEN
       C$ = INPUT$(LOC(1), #1)
       PRINT C$;
       IF respons$ = "" THEN
          respons$ = C$
       ELSEIF (INSTR(C$, CHR$(3)) = 0) THEN
          respons$ = respons$ + C$
       ELSE
          respons$ = respons$ + LEFT$(C$, INSTR(C$, CHR$(3)))
          rrsp = TRUE
          chckrsp = aok(respons$)
          IF chckrsp = FALSE THEN
             badcount = badcount + 1
             IF badcount > 30000 THEN
               badcount = 30000
             END IF
          END IF
       prevrec$ = thisrec$
       thisrec$ = RIGHT$(LEFT$(respons$, 10), 1)
       chkrec = TRUE
       chkrec = rec
       IF chkrec = FALSE THEN
          rcount = rcount + 1
          IF rcount > 30000 THEN
             rcount = 30000
          END IF
       END IF
       respons$ = ""
       END IF
    END IF
  END IF
LOOP
PRINT ""
PRINT ""
IF brc$ = "Y" OR brc$ = "y" THEN
  PRINT "Number of skipped or incorrect responses:  ";
  PRINT STR$(badcount)
  PRINT "Number of non-consecutive record numbers:  ";
  PRINT STR$(rcount)
ELSE
  PRINT "Hit ESC again to return to DOS"
END IF
```

```
CLOSE                              ' End communications.

DO
   KeyInput$ = INKEY$              ' Check the keyboard.

  IF KeyInput$ = CHR$(27) THEN     ' Exit the loop if the user
      EXIT DO                       ' pressed ESC
  END IF

LOOP


END


ticktock:   tick = TRUE
            RETURN

FUNCTION aok (strng$)
aok = TRUE
IF RIGHT$(LEFT$(strng$, 10), 7) = "RF00001" THEN
   firstrec = TRUE
ELSE
   firstrec = FALSE
END IF
IF ((LEN(strng$) <> 45) AND (firstrec = FALSE)) THEN
   aok = FALSE
END IF
IF ((RIGHT$(strng$, 3) <> (CHR$(13) + CHR$(10) + CHR$(3))) AND (firstrec = FALSE)) THEI
   aok = FALSE
END IF
IF RIGHT$(LEFT$(strng$, 6), 3) <> "RF0" THEN
   aok = FALSE
END IF
END FUNCTION

FUNCTION rec

rec = TRUE
IF starting = FALSE THEN
  SELECT CASE thisrec$
     CASE "1" TO "9", "B" TO "F"
        IF (ASC(thisrec$) - ASC(prevrec$)) <> 1 THEN
           rec = FALSE
        END IF
     CASE "0"
        IF prevrec$ <> "F" THEN
           rec = FALSE
        END IF
     CASE "A"
        IF prevrec$ <> "9" THEN
           rec = FALSE
        END IF
  END SELECT
END IF
starting = FALSE
END FUNCTION
```

Parts List - HSBB

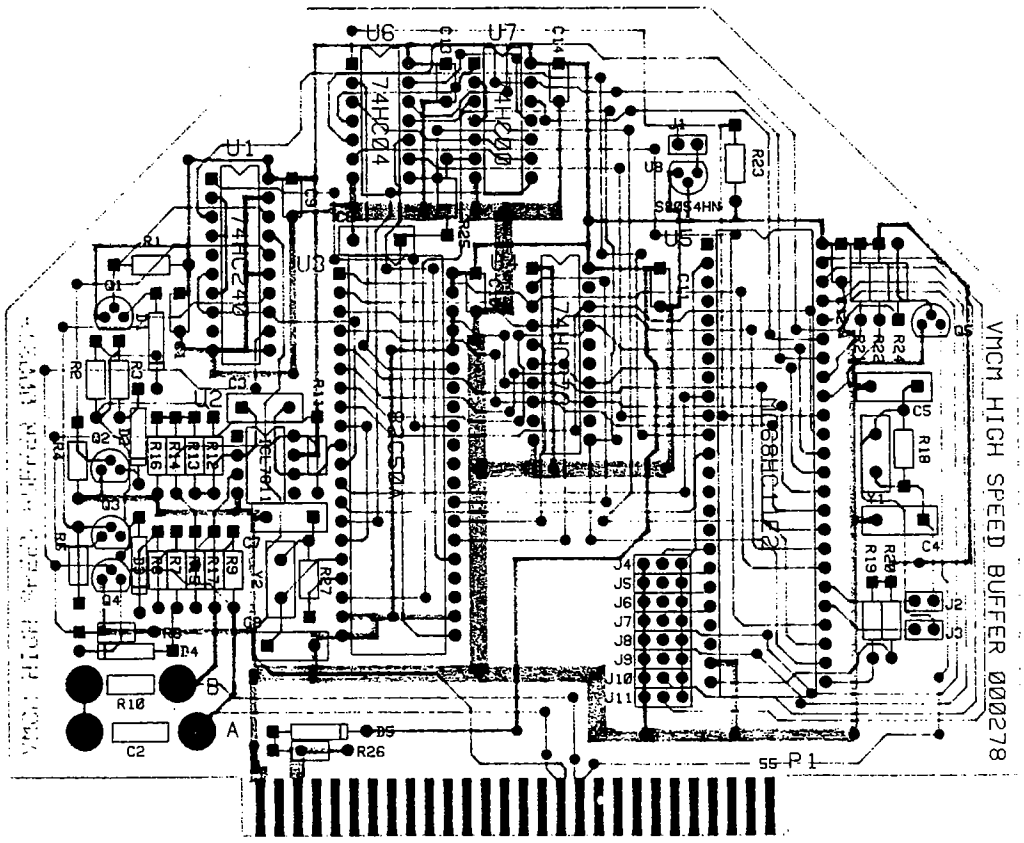
Circuit Element	Quantity	Description	Manufacturer/ Part Number
U1	1	Quad Buffer/Line Driver	MM74HC240N
U2	1	Low Power CMOS Op Amp	ICL7611BCPA
U3	1	CMOS UART/Baud Rate Generator	CP82C50A-5
U4	1	Octal D Latches	MM74HC373N
U5	1	CMOS Microcontroller	MC68HC11A2** **replace with XC68HC811E2
U6	1	Hex Inverters	MM74HC04N
U7	1	Quad 2 Input Nand Gates	MM74HC00N
U8	1	Voltage Detector	Seiko S8054HN
Y1, Y2	2	3.6864MHz Microproc.Crystals	FOX 0368S
D1-D5	5	General Purpose Rectifiers	1N4002
Q1, Q3	2	P channel FET's	Supertex VP0206N3
Q2, Q4	2	N channel FET's	Supertex VN0104N3
Q5	1	NPN transistor	2N3904
R1,R4,R5,R8	4	47 K ohm 1/4 watt 5% carbon film resistors	
R2,R3,R6,R7	4	120 ohm "	
R9	1	12 K ohm "	
R11	1	15 K ohm "	
R12,R17	2	1 Megohm "	
R13,R15	2	120 K ohm "	
R14	1	470 ohm "	
R16	1	20 K ohm "	
R18	1	10 Megohm "	
R19,R20, R21,R22	4	8.2 K ohm "	
R23	1	10 K ohm "	
R24	1	4.7 K ohm "	
R25	1	1.87 K ohm "	
R26	1	10 ohm "	
R27	1	100 ohm "	
C1,C9,C10,C11 C12,C13,C14	7	.1 microfarad ceramic CK05BX104K capacitors	
C4,C5,C6, C7,C8	5	20 picofarad mica CM05ED200J03 capacitors	
J1-J11	11	Jumper headers and jumpers headers: Digikey WM40xx jumpers: AMP SHUNT 531220-3	

Parts List - Resynchronization Module

Quantity	Description	Manufacturer/PartNumber
1	CMOS Dual Monostable Multivibrator	(RCA) CD4098BE
1	P Channel HexFET Transistor	IRF9530
1	1 Pin Connector (Plug and Socket)	Molex
1	24 inches 24 gauge hookup wire	
2	DB-9 Board Mountable Connectors (Plug and Socket)	
1	1.499 Megohm 1/4 watt 1% Resistor	
1	2 microfarad mica capacitor (2 1 microfarad in parallel)	

APPENDIX 4

Layout Drawing



APPENDIX 5

Instructions for assembling, linking, and EEPROM programming.

```
assemble:
            x68c11
                        D
                        CR
                        HSBBFLIP.ASM
                        HSBBFLIP.OBJ

link:       link
                        HSBBFLIP.OBJ
                        F800
                        CR
                        HSBBFLIP.S19
                        CR
                        1

download:
            kermit
                  set port 1
                  set baud 9600
                  connect

power on EVM
hit Master Reset Switch S3
EVMbug Monitor prompt should appear

                  load t
                  Control ] C
                  type hsbb.s19 > com1
                  connect
                  CR

exit kermit:      Control ] C
                  quit

program EEPROM:
```

Use the procedure in the attached pages to erase the EEPROM, then program it, then download again and verify.

3.7.2 Erasing

EEPROM MCU erasing is accomplished by the use of either the BULK or ERASE commands. These commands allows the user to erase memory locations in the MCU internal EEPROM. To perform the EEPROM MCU erasing procedure, perform the following:

a. Place EVM programming switches S4 and S5 to the RST and OFF positions, respectively.

b. Insert MCU device into the EVM programming socket (U32/U56).

c. Apply power via the programmer power switch (S5). Switch is placed to the PWR position.

d. Press EVM MASTER RESET switch S3.

e. Place the programmer reset switch (S4) from the RST (reset) position to the OUT position. This removes the reset condition applied to the MCU, and enables the MCU to be erased.

f. Enter the applicable erase command (BULK or ERASE) via the terminal keyboard. After entering the erase command, the EVM will erase the EEPROM MCU contents within the specified start and ending address.

g. Place programming switches S4 and S5 to the RST and OFF positions, respectively.

h. Remove MCU device from programming socket.

3.6.7 Erase Bytes

ERASE <starting address> [<ending address>]

The ERASE command allows the user to erase individual bytes of the programmed MCU internal EEPROM.

Prior to entering this command, the user must follow the EEPROM erasing procedure as described in paragraph 3.7.2. This procedure removes the reset condition applied to the MCU, and enables the MCU EEPROM to be erased.

The ERASE command is now entered via the terminal keyboard to erase the MCU EEPROM contents. No messages will be displayed on the terminal CRT upon completion of the erase operation, only the EVMbug prompt is displayed.

On MC68HC11A0, A1, and A8 devices byte erasing the configuration (CONFIG) register ($103F) cannot be performed because the CONFIG register is bulk erase only.

EXAMPLES	DESCRIPTION
>ERASE B600 B602 >	Erase MCU EEPROM contents B600 thru B602. Prompt indicates erase sequence completed.
>ERASE B600 B602 M68HC11 NOT BLANK >	Erase MCU EEPROM locations B600 thru B602. Message indicates data stored in specified EEPROM locations is not blank.

3.7.3 Programming

EEPROM MCU programming is accomplished by the use of the PROG command. This command allows the user to program the MCU internal EEPROM. To perform the EEPROM MCU programming procedure, perform the following:

a. Place EVM programming switches S4 and S5 to the RST and OFF positions, respectively.

b. Insert MCU device into the EVM programming socket (U32/U56).

c. Apply power via the programmer power switch (S5). Switch is placed to the PWR position.

d. Press EVM MASTER RESET switch S3.

e. Place the programmer reset switch (S4) from the RST (reset) position to the OUT position. This removes the reset condition applied to the MCU, and enables the MCU to be programmed.

f. Enter PROG command via the terminal keyboard. After entering the PROG command, the EVM will check the EEPROM MCU contents within the specified start and ending address. If any EEPROM MCU locations are not empty, the EVM monitor will prompt the user with a message to either continue the programming sequence or exit the PROG command. A 'RETURN' entry will continue with the programming operation, while any other character will exit the PROG command. This allows changes to be made to already programmed devices. During the programming operation, the terminal CRT display is updated with each address being programmed. Upon completion of the programming sequence, the sequence is automatically verified and the status is displayed on the terminal CRT.

g. Place programming switches S4 and S5 to the RST and OFF positions, respectively.

h. Remove MCU device from programming socket.

EXAMPLES	DESCRIPTION
>PROG B600 B7FF BXXX	Program data from user memory locations B600-B7FF into MCU EEPROM locations B600-B7FF. Display address BXXX updated for each location programmed.
MC68HC11 NOT BLANK ENTER RETURN TO CONTINUE (RETURN) BXXX	Message indicates data is stored at memory locations B600-B7FF. RETURN key depressed to program new data over old data. BXXX is updated for each location programmed.
VERIFY COMPLETE >	Message indicates programming sequence was successful.
>PROG B600 B7FF AA BXXX	Block fill hexadecimal data value $AA into MCU EEPROM locations B600-B7FF. Display address BXXX updated for each location programmed.
MC68HC11 DOES NOT VERIFY >	Message indicates programming sequence did not verify correctly.
VERIFY COMPLETE >	Message indicates programming sequence was successful.

When programming the configuration register a "MC68HC11 DOES NOT VERIFY" message is always displayed on the terminal CRT. This is because the configuration register cannot be read without resetting the MCU. To check proper programming of the configuration register, first reset the MCU via the programmer reset switch S4, and then copy the location into RAM. The configuration register is then verified visually.

>PROG 103F 103F 0E 103F	Program hexadecimal data value $0E into MCU configuration register at location $103F.
MC68HC11 DOES NOT VERIFY >	Message indicates programming sequence did not verify correctly.

DOCUMENT LIBRARY

January 17, 1990

Distribution List for Technical Report Exchange

Attn: Stella Sanchez-Wade
Documents Section
Scripps Institution of Oceanography
Library, Mail Code C-075C
La Jolla, CA 92093

Hancock Library of Biology &
Oceanography
Alan Hancock Laboratory
University of Southern California
University Park
Los Angeles, CA 90089-0371

Gifts & Exchanges
Library
Bedford Institute of Oceanography
P.O. Box 1006
Dartmouth, NS, B2Y 4A2, CANADA

Office of the International
Ice Patrol
c/o Coast Guard R & D Center
Avery Point
Groton, CT 06340

NOAA/EDIS Miami Library Center
4301 Rickenbacker Causeway
Miami, FL 33149

Library
Skidaway Institute of Oceanography
P.O. Box 13687
Savannah, GA 31416

Institute of Geophysics
University of Hawaii
Library Room 252
2525 Correa Road
Honolulu, HI 96822

Marine Resources Information Center
Building E38-320
MIT
Cambridge, MA 02139

Library
Lamont-Doherty Geological
Observatory
Columbia University
Palisades, NY 10964

Library
Serials Department
Oregon State University
Corvallis, OR 97331

Pell Marine Science Library
University of Rhode Island
Narragansett Bay Campus
Narragansett, RI 02882

Working Collection
Texas A&M University
Dept. of Oceanography
College Station, TX 77843

Library
Virginia Institute of Marine Science
Gloucester Point, VA 23062

Fisheries-Oceanography Library
151 Oceanography Teaching Bldg.
University of Washington
Seattle, WA 98195

Library
R.S.M.A.S.
University of Miami
4600 Rickenbacker Causeway
Miami, FL 33149

Maury Oceanographic Library
Naval Oceanographic Office
Stennis Space Center
NSTL, MS 39522-5001

Marine Sciences Collection
Mayaguez Campus Library
University of Puerto Rico
Mayagues, Puerto Rico 00708

Library
Institute of Oceanographic Sciences
Deacon Laboratory
Wormley, Godalming
Surrey GU8 5UB
UNITED KINGDOM

The Librarian
CSIRO Marine Laboratories
G.P.O. Box 1538
Hobart, Tasmania
AUSTRALIA 7001

Library
Proudman Oceanographic Laboratory
Bidston Observatory
Birkenhead
Merseyside L43 7 RA
UNITED KINGDOM

50272-101

REPORT DOCUMENTATION PAGE	1. REPORT NO. WHOI-90-33	2.	3. Recipient's Accession No.

4. Title and Subtitle

The High Speed Buffer Board – A SAIL EIA-485 Communications Accelerator Card for the Vector Measuring Current Meter

5. Report Date
July, 1990

6.

7. Author(s)
Robin Singer and Douglas M. Butler

8. Performing Organization Rept. No.
WHOI 90-33

9. Performing Organization Name and Address

The Woods Hole Oceanographic Institution
Woods Hole, Massachusetts 02543

10. Project/Task/Work Unit No.

11. Contract(C) or Grant(G) No.
(C) N00014-84-C-0134
(G) N00014-90-J-1495

12. Sponsoring Organization Name and Address

Funding was provided by the Office of Naval Research

13. Type of Report & Period Covered
Technical Report

14.

15. Supplementary Notes

This report should be cited as: Woods Hole Oceanog. Inst. Tech. Rept., WHOI-90-33.

16. Abstract (Limit: 200 words)

A High Speed Buffer Board (HSBB) has been developed for the Vector Measuring Current Meter (VMCM) to implement the transmission of data at 9600 baud over an EIA-485 link. The HSBB significantly extends the VMCM communication functionality, which was previously limited to 300 baud transmission via 20 mA current loop or FSK telemetry. The increased speed allows rapid sampling of a large number of current meters on a common cable and the EIA-485 circuitry, which was designed for low power operation, provides a useful multipoint communication method for data transmission over long cable lengths. SAIL protocol (IEEE 997) was utilized to coordinate data transfer by the instruments on a common link.

An MC68HC11 microcontroller resides in the VMCM, buffering data it receives at 300 baud from the VMCM UART. In response to a jumper selectable SAIL address, the MC68HC11 offloads the data at 9600 baud via EIA-485 to the SAIL controller. Synchronous data collection from many instruments is ensured by the SAIL synoptic set command and an embedded resynchronization/reset command. The low power consumption allows deployments of six months or more with a standard VMCM battery stack.

17. Document Analysis **a. Descriptors**

1. current meter
2. telemetry
3. EIA-485

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

18. Availability Statement

Approved for publication; distribution unlimited.

19. Security Class (This Report)
UNCLASSIFIED

20. Security Class (This Page)

21. No. of Pages
45

22. Price

(See ANSI-Z39.18) *See Instructions on Reverse* OPTIONAL FORM 272 (4-77)
(Formerly NTIS-35)
Department of Commerce