Hydraulics Research
Wallingford

MEASUREMENTS OF WAVE DIRECTIONAL SPECTRA AT SEA - A wave direction processor

M Towers BA

*CORRECTED BY MIKE

Report No. SR 16
January 1985

Registered Office: Hydraulics Research Limited,
Wallingford, Oxfordshire OX10 8BA.
Telephone: 0491 35381. Telex: 848552

This report describes work carried out under Contract DGR/465/32, funded by the Department of Transport from April 1982 to March 1984 and thereafter by the Department of the Environment. Any opinons expressed in this report are not necessarily those of the funding Departments. The DoE (ESPU) nominated officer was Mr A J M Harrison. The work was carried out by Mr M Towers in the Technical Services Department of Hydraulics Research, Wallingford, under the management of Dr A J Brewer. This report is published with the permission of the Department of the Environment.

CONTENTS

APPENDIX A

Specification of field processor for proposed velocity/pressure gauge.

ABSTRACT

The development of the microprocessor system described in this report was prompted by an evaluation made by Hydraulics Research of methods for the measurement of wave directional spectra at sea. This earlier study suggested the need for a low-powered processing unit suitable for incorporation in a compact field gauge recording horizontal water motions and depth fluctuations at the sea-bed. The operational requirement is that, given the signals from the velocity and pressure sensors, the microprocessor calculates the necessary parameters for describing the directional properties of the waves. Storage of the descriptive parameters instead of the raw velocity and pressure data drastically reduces the quantity of information that has to be logged and hence offers prospect for gauge deployments of several months duration.

The present report outlines the reasons for basing the development on the 16-bit Texas TMS 9995 microprocessor; the circuit and software design; and initial testing of the processing routine using simulated signal inputs. The present stage is only one in the development of an overall recording system that will find application whenever data on wave directions have to be collected, provided that water depths are not too great to prevent the influence of surface waves penetrating to a measurable degree at the sea-bed. It follows that it will be particularly valuable for studies of bottom water processes such as sediment transport under waves or for problems associated with submarine pipelines and platforms. The choice of suitable velocity and pressure sensors together with the design of low-powered driving circuits, data quality monitoring, general system control and the recording of the processed data will be the subjects of future work on the overall project.

1. INTRODUCTION

A review (Ref.1) undertaken by Hydraulics Research in 1983 into methods for measuring wave directional spectra at sea recommended the development of a self-contained sea-bed recording gauge. The requirement is for a compact device that can be deployed simply from a small fishing vessel or launch without any need for elaborate mooring provision. The small wave orbital buoy made by the American Company Endeco Inc was a possible candidate to meet this need. However in a field trial of that device, fully reported in Ref.1, the Endeco buoy failed to produce sensible directional data. On the same trial a make-shift velocity and pressure gauge laid on the sea-bed yielded consistently credible records. This promising result justified the injection of further funds into the alternative of a sea-bed gauge. The aim was to extend the time between servicing such a gauge to three months or more.

The principle of extracting wave directional spectra from simultaneously measured horizontal water motions and water depth at the sea-bed has been employed elsewhere. Aubrey (Ref.2) has evaluated the performance of a sea-bed gauge incorporating a two-component electromagnetic current meter and pressure sensor. In this instance each record comprised 2048 scans at one second intervals with three readings per scan being written to cassette tape with unusually high density of data packing. In common with other deployments of similar gauges the logged data is returned to a shore base for fast fourier transform (FFT) analysis to derive the various descriptive parameters of the sea.

The large quantities of data produced when the velocity and depth signals are sampled and stored directly limits severely the time such a device can be left between tape changes. Even with the data compaction obtainable on the cassette logger used by Aubrey the deployment time was restricted to about six weeks with 34 minute records every six hours, a total of 180 records. The objective of the present project is to reduce the data storage demand by performing the spectral analysis in situ and recording only the results. These are considerably more compact than the raw data. The general specification for the required gauge was given as an appendix to Ref.1 and for convenience it is repeated in Appendix A. In brief the minimum requirement is to achieve 34-minute records every three hours throughout three months, or a total of 720 records.

The first stage of the project is the development of a microprocessor-based circuit capable of undertaking data analysis within the sea-bed package. Software for executing the numerical data analysis already exists in the form of a main-frame Fortran program but needs to be adapted so that the computation is made on a suitable microprocessor of low power consumption. It has been decided to split the functions of data acquisition and data processing between two microprocessor subsystems. One will be concerned with handling the various peripherals such as controlling the operation of the transducers and recording of results together with system and data checking. The second will be dedicated to the numerical processing function. The present report is primarily concerned with the development of the latter although some leads are given for the implementation of the data acquisition subsystem which will be the subject of future work.

2. DESIGN CONSIDERATIONS

The location of the logger on the sea-bed means that it must carry its own power supply and low power consumption becomes an important design criterion. However the numerical processor need only be turned on when there is data to be analysed, so its working current consumption is not too important, if it is only active for a small fraction of the time. This allows greater freedom in its design than if a low working power consumption was vital. Another important consideration is how the software will be developed; this is likely to present the greatest cost factor in the design of the system. Finally the processing requirements themselves will place some restrictions on both the hardware and software which can be considered for the job.

2.1 General processing requirements

The data to be processed are three channels (two velocity components and one pressure) of 4096 values, each value occupying perhaps twelve bits. These values are probably most easily handled as sixteen-bit integers, which would imply a storage requirement of 24 Kbytes. The processing is likely to be conceptually simpler if floating point representation is adopted for calculations, thus effectively eliminating the concern about overflow and loss of precision which would be a concomitant of selection of a fixed point representation. This suggests that four bytes per value would be optimal, using a standard floating point format. During the calculations the largest structure will be the array of 2048 complex numbers on which the Fourier Transform is performed (and each channel must be so treated). This requires 16 Kbytes. Certain data structures are used by the transformation process (to increase speed); these are an array of complex roots of 1 (8 Kbytes) and an array of bit-reversed integers (1 Kbyte). It can be seen that memory requirements are fairly large and data overlaying should be given considerable attention: it is particularly desirable that the total memory requirement, including code, should not exceed 64 Kbytes, which is the largest quantity small processors can comfortably handle. It was decided that, by

by transforming each channel separately and then compressing the resulting array (16K) back into the array in which the data was supplied (8K), it could be arranged that (a) all three channels could be loaded at the same time, (b) no significant loss of precision would occur, and (c) total data memory requirement would be less than 56K, leaving at least 8K for code.

2.2 Choice of programming language

The most obvious approach to developing a system of this complexity is to write the software in a high level language on an existing computer, where it may be tested and debugged, then recompile it for the actual application processor. It is then desirable that the end product be designed so that on-line debugging may be performed, with the code actually running in its intended environment (the environment in the end product is never the same as that in the development machine). This approach has not been used in this project; the software has been written in assembly code and can only be tested in its final environment. The reasons, not particularly in order of importance, are:

(a) absence of any high level language development system;

(b) speed, the faster the operations can be performed the shorter the time for which the processor must be activated, hence lower power;

(c) ease of communication, using a high level language makes coding the processing routines much easier, but interfacing to it may be a problem;

(d) acquaintance with the hardware and assembler;

(e) the only other available choices BCPL or, perhaps, BASIC, were rejected on the grounds that they would both have to be run interpretatively which would mean low speed and also the space taken by the interpreter would probably be unacceptable. Not only this but the representations of numbers in BBC BASIC (the only type readily supported) are rather large, four bytes for an integer and five for a floating point number, again a space problem.

The disadvantages of the chosen approach are that all the arithmetic routines (addition, multiplication, trig. functions etc) must be written, the resulting programs are not easy to read, and debugging is generally difficult. The unreadability of programs is an important failing and the approach adopted to alleviate this problem is to separate the program into manageably small, logically distinct modules. High level descriptions of each module must be written and details of the mapping to assembly code must be provided. This also aids debugging.

3. HARDWARE

Given the absence of a high level language development system and the consequent need to write software in assembly language a major factor in the choice of hardware configuration was the complexity of the instruction set of the processor. The ease of design of a comprehensive on-line debugging tool was also important. The complicated processes needed to manipulate 16 bit quantities in 8-bit microprocessors led to their rejection; the software development would have been complex and the lower power consumption which would have resulted from using CMOS circuitry would have been at least partially offset by the increased computation time. No 16-bit microprocessors are currently readily available in CMOS so standard NMOS/LSTTL circuitry is used. The choice of TMS9995 microprocessor was made for the following reasons:

(a) it is the simplest existing 16-bit processor, allowing easy development of software;

(b) it needs fewer support chips than other microprocessors;

(c) it allows fairly straightforward development of the hardware for adequate on-line debugging facilites;

(d) a design for a suitable processing board was available.

The hardware is essentially a 16-bit microprocessor with 64 Kbytes of dynamic RAM, of which either 8K or 16K may be replaced by EPROM. All of this memory is accessible by an external device, which also handles the processor's control lines (reset, interrupts etc.). This arrangement makes design of a debugging system quite simple. This interface may also be used by the data acquistion part of the logging system to activate and communicate with this numerical processor.

The 16 bit processor will normally be off. When a set of data is available, the numerical processor may be activated. The data will then be transferred to it and processing can be started. When analysis of the data is finished the results must be sent to a tape (or other storage medium). The medium will be another peripheral of the data acquisition processor, so the results will be transferred back to it. This leaves the numeric processor design almost totally independent of the rest of the system so it can be developed without worrying unduly about the nature of the data collection and storage processes.

4. DEVELOPMENT FACILITIES

The software has been developed on a BBC micro, using the TMS9995 processor board itself for testing and debugging, under the control of an adaptable BBC program Although this program is not very comprehensive in the facilities it offers it is written in BASIC and can be easily modified. The items used in the development process will be listed together with descriptions of their functions within this context.

4.1 BBC micro

Used for the production of application code, running debugging aids, controlling the TMS9995 processor board, and writing documentation. It requires at least one 40-track disk drive and the BCPL package.

4.2 Interface between BBC micro and TMS9995 processor board

This interface converts the signals from the BBC's User Port to a form suitable for driving the processor board.

4.3 Text editor

The text editor is used for (a) entry and modification of applications code. For this purpose the "ED" editor supplied with the BCPL package has been used. (b) Entry and modification of development and debugging tools. Those written in BASIC have just used the standard built in commands; those written in BCPL have used ED. (c) Documentation has used Wordwise. Wordwise and ED could each have performed all the editing tasks but since they are both available the choice has been governed by their relative merits in the various situations.

4.4 ASM9995

This is the TMS9995 assembler, developed in BCPL on the BBC micro. It produces relocatable object code from a source file of assembly code text, and is invoked by a command of the form:

ASM9995 ⟨source file⟩ (object file)(/L) (NOLIST)

where ⟨... file⟩ is a BCPL system compatible filename (e.g. A.WD2 or /F.0.WD6) and (..) denotes an optional parameter, /L for a listing to the printer and NOLIST to suppress listing, except of errors. See the relevant file (TMS9995 Assembler) for further details, including source code.

4.5 BLINK

This links relocatable object code files such as are created by ASM9995 to produce larger relocatable object files. This program has also been written in BCPL on the BBC micro (there is a file containing its description and source code).

4.6 WDL

This loads a relocatable object file to the TMS9995 processor board. It has been written in BCPL and 6502 assembler on the BBC micro specifically for this project. It can handle segment names "CODE", "DATA", RESET" and "FPP". The addresses to which the segments are loaded are set in module WDL2 (source in B.WDL2), procedure FINDSTART. The component files are (a) B.LDØ, the control directing module. (c) B.WDL1, generally overseeing the process. (d) B.WDL2, containing various auxiliary routines. (e) LHDR, the header file. It is run by issuing the command WDL. It will then ask whether a display of global addresses is required and prompt for the file to be loaded. It leaves the processor board in a held state, having previously issued a reset command. If a listing of the global addresses to the printer is required, press control-B before running the program.

4.7 WDD

This is the debugging and testing program, written in BASIC. Parts of it are dependent on the code loaded to the procesor board; it uses various addresses in the code (such as that of N, the number of points in the transform), and the breakpoint function will only work if the appropriate code fragments are present (these currently appear in module WDØ). It is menu driven so operation is quite simple. At present it

includes code for the supply of a mixture of up to 10 sinewaves as data to the processor; the generation and loading of this data can be rather a lengthy process (nearly five minutes for one sinewave with N=2048, i.e. 4096 points). Note that BASIC must be entered before this program can be run (all others run in the BCPL system).

4.8 WDPL

This a a variation of WDL for loading the code to an 8K file, rather than the processor board, so that it may be written to an EPROM. It runs similarly, except that it asks for the name of an output file. Its constituent modules are B.WDPLO, B.WDPL2 and LPHDR.

4.9 EPROM programmer

To program an EPROM using the output file from WDPL, when the code has been finalised. This EPROM will occupy the same space in the processor as the code did during debugging and should run identically. A couple of links on the processor board must be changed to incorporate the EPROM within the memory map. The programmer used is the Watford Electronics model, which is controlled by a BBC micro (in this case not the same one as used for development work).

5. EXAMPLE SHOWING USE OF DEVELOPMENT SYSTEM

By way of illustration of the interrelationship of the component parts of the development system, the steps involved in modifying a source file (e.g. A.WD4) and testing the resulting code will be outlined.

5.1 Edit source file

Must be in BCPL system to use ED. Enter TMS9995 code disk into appropriate drive.

```
ED A.WD4
```

The alteration may now be made (see BCPL manual for operation of ED editor) and on exit the new file will be in store. Save it to disk!

```
SAVE A.WD4
```

5.2 Assemble source file

Must be in BCPL system. Use TMS9995 code disk again.

```
ASM9995 A.WD4 O.WD4 NOLIST
```

This assembles the file to store file O.WD4 (again this should be saved!). The NOLIST option is used so that errors will be clearly displayed. If a listing is required a further pass of the assembler is recommended, to avoid producing pages of listing when there are still errors in the source.

```
ASM9995 A.WD4/N/L
```

This time the null file is used for output since we have already obtained the object file.

Notes

(a) In addition to saving the object file on the present disk it should be saved on the "object code linkage" disk.

(b) It may be that an object file may not fit into store whilst its source file and the assembler are resident. In this case it must be assembled directly to disk. A.WD6 has this attribute.

```
ASM9995 A.WD6 /F.O.WD6 NOLIST
```

5.3 Object code linkage

Must be in BCPL system. Use "object code linkage" disk.

```
EX LNK
```

This saves typing the complete list of object files every time the linkage is performed. About half way through a request for a list of names is issued (symbols for suppression). If the names of any globals are entered here they will not be passed to the output file; this feature is unlikely to be useful in this application, just press "return". The name of the output file will then be requested. This has, to date, been O.WD, but note that its size is such that it is not possible to create it in store so "/F.O.WD" must be typed. This file may be transferred to the "BBC end" disk for loading.

5.4 Loading the code

Must be in BCPL system. Use "BBC end" disk.

```
WDL
```

A question will be displayed, "List global addresses?". For debugging purposes it is very helpful if the machine addresses of those items declared as global are known. An affirmative answer to this question will supply this list. Note that if a listing to the printer was required, control-B should have been pressed before running this program. Press "break" and start again. The next request is for the object file name, and when this has been supplied the program will run to completion, leaving the processor in a "held" state. For the loading to be carried out successfully the BBC micro should be connected to the processor board via the interface and the power supply to the latter two units should be active.

5.5 Testing the code

First enter BASIC. Use "BBC end" disk.

```
CH."WDD"
```

The steps in a simple test run will now be listed.

(a) Set the number of data points (2N), to e.g. 1024.

(b) Load the data. A request for frequency, amplitude and direction for each of up to 10 components is issued. If a null response (just "return") is given to a frequency request it will terminate the list. A 1024 point single sinewave takes about a minute to load.

(c) Run the program. "Normal completion" will appear if it runs to completion. If this does not happen then the breakpoint facility may be used, in conjunction with that for memory access, to trace the problem (which, if only one modification has been made, is likely to be closely associated with that change).

(d) Display the results. This routine displays (optionally to the printer) the two estimates of spectral density and, calculated from ALPHA1, the wave directions and spreads. If these do not meet with expectations, the algorithms and code must be inspected, perhaps using breakpoints to investigate unexpected code behaviour.

5.6 Putting the code in EPROM

The final location of the code developed in this project is an EPROM, which may be plugged into the appropriate socket on the processor board, so that it will run immediately the processor is activated. The EPROM programmer requires an 8K file matching the proposed contents of the EPROM (a 2764), so a program has been written (WDPL, on disk "EPROM loader") to convert the object file to this form. It is very similar in function and design to WDL (not surprisingly since the only functional difference is the destination of the output). The disk containing the output file is then transferred to the machine driving the EPROM programmer and the fairly simple menu driven software supplied with the programmer is used to program the EPROM (see EPROM programmer manual for details).

6. COMMENTS ON DEVELOPMENT SYSTEM

1. When using an EPROM for program storage, WDD will not be able to change the number of points, so if fewer than 4096 samples are wanted a new EPROM must be prepared. The program (WDO) might be modified to allow new values to be set, perhaps by stopping after initialisation.

2. If it is desired that the data acquisition processor sets any of the constants listed in WDO then WDO will have to be modified so that this value is not initialised internally. These values will in any case have to be modified to suit the actual experimental conditions. No changes to other modules should be necessary in this context.

3. Any constants that do have to be sent to the numeric processor by the controller might best be sent in the form in which they arise in that processor, leaving conversion to floating point to the numeric processor. The extra code, presumably placed in WDO, would not be extensive (an integer to floating point conversion routine is already available as S.FLT, in module MISC of the floating point package).

4. For maximum speed the registers should be located in internal RAM, by changing the contents of absolute address 0 (beginning of segment RESET) from F100 to F000, either by modifying WDO or by changing this value when transferring the code to EPROM.

7. SOFTWARE

Section WD1 performs d.c. blocking (by subtracting the mean from all data values) and Blackman windowing on 2N integer data values, leaving the result in floating point format in the array CDAT. Section WD2 sets up arrays IR and FI using the FFT process. The first stage of the Fourier Transform is in WD3 (transforms N complex points). The second stage, to construct from the result of WD3 the transform of 2N real points, is in WD4. The data at this stage occupies 16K (in CDAT) and if it were to be retained in this form the space taken by the three channels would be 48K. This is too much: [CODE (8K) + FI (8K) + IR (1K) + Miscellaneous data + 48K]= >64K], and this is with much data overlaying. There are a number of possible solutions to this problem, the one used having the merit of leaving the algorithm essentially untouched; it involves compressing the floating point representation to an integer representation (not twos complement) which occupies half as much space, allowing the array to be placed in the same space in which the data samples were first supplied. This compression is performed by CMPRS in WD5. These compressed arrays are called AA, BB and CC. The memory allocation may now be sketched.

This is no longer the case. The frequency range of interest is all within the bottom half of the processed band so the necessary compression is simply achieved by discarding the top half of each array.

```
0000-1FFF  :  code
2000-3FFF  : FI
4000-5FFF  : DAT (pressure) / AA
6000-7FFF  : DAT (velocity 1) / BB
8000-9FFF  : DAT (velocity 2) / CC
A000-DFFF  : CDAT
A000-?     : S33
A800-?     : S44
B000-?     : ALPHA1
B800-?     : ALPHA2
E000-E3FF  : IR
F200- FFF0 : small bits of data, segment
DATA
```

The steps outlined above are performed for each of the three channels, CDAT being common workspace.

The final stages are performed by code in WD6 using functions WAVEND (wavenumber calculation) and XPND (for conversion from the integer format in AA, BB and CC to floating point format) from WD5 and also the short procedures in WD7. WD7 really owes its existence to the limitations of the ED editor on the BBC micro: the "Fx" procedures are closely associated with the code in WD6 and would have been included in WD6 if the editor could cope with such a large file that would then have resulted.

The purpose of this last section is to reduce the transformed data, in AA, BB and CC to 0.01 Hz frequency bands, using auto- and cross-spectra to produce energy density spectra and the quantities "alpha 1" and "alpha 2" described in Ref.1.

The sequencing of operations, initialisation, provision of constants, etc. is handled by WD0. In its basic form it is fairly passive, requiring the data to be present before it runs and leaving the output arrays waiting to be read. Exceptions, completion etc. are signalled by interrupting the host having left clues to the reason for the prod lying around in memory. This is fine for a host which implements the full features of the interface to the host in the final logging application, if only to reduce the number of wires. Extra communication code would then be necessary. This will be considered when the requirements of the host processor itself is studied.

seems to be a bit missing

8. TEST RESULTS

Sinewaves have been generated using the routine in WDD, which allows choice of frequency, amplitude and direction for a set of sinewaves, the sum being entered into the processor board data arrays.

The time taken for the complete processing operation is just under two minutes. The following breakdown giving approximate times shows where the bulk of the workload lies.

1) preliminaries, including generation of F1 and IR: 2 seconds;

2) PREPROC: 52 seconds. This may be further broken down into:

   (a) BLW: 14 seconds. includes integer to floating point conversion and Blackman windowing;

   (b) FFT: 32 seconds. Complex Fast Fourier Transform. The time taken by SHUFFLE is less than 1 second.

   (c) RECON: 8 seconds. Conversion of transform coefficients for real data;

   (d) CMRPS: 1 second. Conversion of floating point transform coefficients to scaled integers.

   This group of procedures is performed three times, once for each channel.

3) FINALE: 8 seconds. Auto- and cross-spectra and compression of the data into wider frequency bands, followed by calculation of output arrays.

The above totals nearly 3 minutes but this applies to the tests being made with registers in external RAM. Although during testing and debugging the TMS 9995 register space is located in external RAM in order to be accessed by the BBC micro, a considerable increase in speed is obtainable with the registers in internal RAM as will be the case with the program fully operational. The corresponding times are:

1. Preliminaries: 1 second

2. PREPROC: 36 seconds

   (a) BLW: 8 seconds
   (b) FFT: 21 seconds
   (c) RECON: 5 seconds
   (d) CMPRS: 1 second

3. FINALE: 4 seconds

Clearly the FFT is the most time consuming procedure, taking more than half the total time. However, further tests, involving repeated break-in's suggest that more than two thirds of the time is spent performing floating point addition/ subtraction (alone nearly half the time) and multiplication, routines which are used by most parts of the program, especially FFT. The routines for complex addition/subtraction and multiplication (these call the appropriate floating point routines) also occurred fairly frequently, limiting all other program activities to less than 10% of the time. Obviously any improvements would have to lie in the elementary arithmetic routines. Additional hardware, in the form of a floating point processor, might be considered but is presently impractical, there being nothing suitable available. Alternatively a fixed point representation could be adopted. This would be possible to implement but is not recommended for several reasons:

(a) very considerable modifications to the software would be necessary;

(b) the wide range of values occurring at different stages of computation might mean that different fixed point representations would be necessary to retain precision, resulting in the need for renormalisation procedures;

(c) the possible renormalisation and range checks would obscure the program flow, making it difficult to follow and hence error prone. However, if an increase of speed prove vital, such modifications might halve the processing time although the necessary checks might reduce the gain.

In practice the overall power consumption of the numeric processor is much less than the expected power demand of the transducers. Operational current is about 0.6A, which for two minutes per record and 720 records amounts to 14.4Ah. A supply voltage to the on-board regulator of 7 to 8V is recommended.

9. THE DATA ACQUISITION PROCESSOR

The companion microprocessor dealing with data acquisition must control the transducers, the storage device for the results of the spectral analysis, and the interface to the numeric processor. In addition it will have to perform data quality checks, though if the processing involved in this activity becomes substantial consideration should be given to transfering this function to the numeric processor.

Regard should be taken of the following points when the project enters onto the development of the data acquisition subsystem:

1. The NSC800 or possibly 65C02 are likely to be suitable processors. Assemblers are available for both these devices, though that for the NSC800 is perhaps more convenient, being close in implementation to that for the TMS9995, in particular allowing use of the same linker and very nearly the same loader.

2. In the interests of low total power consumption it is desirable to:

   a) use CMOS circuitry;

   b) employ a low clock speed;

   c) deactivate as much circuitry as possible when it is not in use. However, with the processor off between records it will be necessary to have some memory kept active if any information has to be retained from one record to the next;

   d) possibly employ a processor power-save mode between samples, which are, in computer terms, quite well separated (½ second).

3. The processor should be completely reset for every record, to protect against crashes.

10. INTERFACING THE TWO PROCESSORS

When the complete logging device is assembled, the unit performing data acquisition and storage must take over the control of the numeric processor board, a task which had hitherto been in the domain of the development system. The steps it will have to go through, for each set of data are:

1. Collect 2N samples from each of the three channels.

2. Activate the numeric processor.

3. Reset and "hold" it.

4. Transfer the samples to their allotted locations in the numeric processor's memory.

5. Start the numeric processor ("unhold" it).

6. Wait for completion to be signalled.

7. Read back the results from the numeric processor's memory.

8. Deactivate the numeric processor.

9. Save the results to tape, or whatever other medium is selected for storage of results.

The above only gives an outline of its duties; housekeeping information must be saved with the table of results, data errors must be handled, and appropriate action must be taken if the numeric processor signals an error.

The simplest approach to the interface would be to leave it as it is, requiring the new controlling processor to supply similar signals and access sequences to those used by the development system. The hardware for this approach would not be extensive; a parallel interface chip could provide all the data and control lines, perhaps less than 16 being necessary. No additional software for the numeric processor board would be required and there is every prospect that only simple software would be needed for the controller. It is recommended that this straightforward approach is adopted. However, if unforeseen complications arise a possible alternative path is outlined below.

In the simple method the controller must, in addition to supplying the data to be transferred between memory spaces, supply the relevant address in the numeric processor for each byte. Now, it is desirable that (for power economy) the numeric processor be active for as short a period as possible, so if the data transfer process occupies a significant fraction of the total active time it would be beneficial to attempt to reduce it. It must be said that this does not seem likely but if the controlling processor is running very slowly (to minimise power consumption) it might happen. Thus it could help if the task of address handling were removed from the controller, leaving it with only data bytes to handle. This would require interaction with software in the numeric processor and a communication protocol. This might work as follows:

1. Activate, reset and run numeric processor, which will wait for a command from the controller.

2. Send a (series of) byte(s) describing operation required (read or write), length of block, and start address.

3. Read or write said number of bytes.

The hardware might be arranged such that the controller only has access to one byte in the numeric processor memory. Hardware must be present to adjust the numeric processor interface control lines (since software would be slow, defeating the object of the exercise); here "hold" must be activated and, possibly, READY awaited for each byte transfer, since the numeric processor is not continuously held as it was in the simple interfacing method. In addition a signal must be sent to the numeric processor, presumably in the form of an interrupt, to indicate that the transfer has occurred. The relative speeds of the processors (which is likely to be great if this method is used) may mean that no handshake signal is needed by the software in the controller (carrying on at full speed secure in the knowledge that all necessary action will have been taken by the numeric processor before the slow process gets round to the next byte). The interactions between the processors may be so static that the communication protocol can be quite trivial, perhaps just numbered interactions.

The controller software for this method may well be even simpler than in the first method (it must run faster). However the price paid is extra software in the numeric processor which must now handle the location of data within its memory space and the loss of generality of the interface.

11. CONCLUSIONS

A dedicated processor board using the Texas TMS 9995 microprocessor has been developed and a printed circuit has been produced. The circuit has been superficially tested using a simulation method to prove the principle spectral analysis and direction extraction functions. The design of the development tools, the reasoning exercised over the choice of devices, and the software have been described. The detailed software design and the listings are not included with this report, as the resulting documentation is far too bulky. These together with the microprocessor subsystem and associated circuit diagrams are held at Hydraulics Research Ltd, Wallingford.

The next step in the development program will be the selection of suitable transducers paying due regard to the need for low power consuming devices. The acquisition of transducers and the design of power supply driving cirucits are expected to be undertaken in the coming year with funding by The Department of the Environment. Then the project will enter its final phase with the development of the data acquisition microprocessor and the integration of the various subsystems in a submersible housing. The project is expected to run for another two years.

12. ACKNOWLEDGEMENTS

This work was carried out by Mr M Towers under the supervision of Mr I E Shepherd in the Technical Services Department headed by Dr A J Brewer.

13. REFERENCES

1. M J CRICKMORE and R W P MAY. Evaluation of a wave orbital buoy for measuring directional spectra at sea. Hydraulics Research Report IT 256, 1983.

2. D G AUBREY. Field evaluation of Sea Data directional wave gauge (model 635-9). Woods Hole Oceanographic Institution Tech.Report WHOI-81-28, 1981.

APPENDIX A

Specification of Field Processor for proposed Velocity/Pressure Gauge

The objective is to develop a battery-operated, low power field microprocessor to compute directional wave spectra within a self-contained velocity/pressure gauge. Its functions are to include:

1. the control of individual sensor operation to permit a repeating sequence of sensor power on, record start, record finish/power off, and rest period of selected duration (e.g. 1, 3 or 6 hours);

2. the acceptance in the recording phase of continuous signals of pressure and of two velocity components, together with a single reading per record of magnetic heading. Data quality checks are to be made on the input signals;

3. the computation by FFT techniques of the directional spectrum of each wave record and making the spectral data available for digital recording in a suitable medium for subsequent recovery.

Input

The sensors that are to make up the front end of the velocity/pressure gauge have not been decided finally but present expectations are that they will comprise:

(a) 100mm diameter Colnbrook electromagnetic current meter

(b) Marinex flux gate compass

(c) Paroscientific Digiquartz pressure transducer.

The outputs of (a) and (b) are analogue whereas (c) is digital. The target reading resolutions are 10 mms^{-1} over a horizontal velocity range of $\pm 3ms^{-1}$; one degree; and 1 millibar over a range 1 to 4 bar (i.e. to 30m water depth).

Signal checks

Suspect readings are to be identified in the following classes:

(a) zeroes

(b) out-of-scale

(c) excess gradient between successive readings

(d) persistent no change in reading

Operation

The standard operating sequence will be to record for about 34 minutes every 3 hours although 1 and 6 hours intervals are desirable options. Spectral analysis based on 4096 samples (i.e. 2Hz sampling) is preferred but could be relaxed to 2048 (1 Hz sampling) if considerable design advantage is obtained. Unattended operation of at least 3 months is desirable i.e. 700 to 750 individual records.

Spectral analysis is to be based on 0.01Hz frequency bands covering the range 0.04 to 0.3Hz.

Output

Values for each 0.01Hz frequency band specifying energy density to the nearest $0.01 m^2 s$, mean direction and effective spread to the nearest degree are to be available for digital recording or storage.

Each record output is also to contain information on data quality checks and general housekeeping e.g. the number and class of suspect and inserted readings on each of the three raw data channels; power supply voltage; real or elapsed time.