I.O.S.

ROUTINE STORM SURGE FORECASTING USING NUMERICAL MODELS: PROCEDURES AND COMPUTER PROGRAMS FOR USE ON THE CDC CYBER 205E AT THE BRITISH METEOROLOGICAL OFFICE

BY

R. PROCTOR AND R.A. FLATHER

REPORT NO. 167

1983

NATURAL ENVIRONMENT RESEARCH COUNCIL

INSTITUTE OF OCEANOGRAPHIC SCIENCES

When citing this document in a bibliography the reference should be given as

PROCTOR, R. & FLATHER, R.A. 1983 Routine storm surge forecasting using numerical models: procedures and computer programs for use on the CDC CYBER 205E at the British Meteorological Office. *Institute of Oceanographic Sciences, Report*, No.167, 171pp.

INSTITUTE OF OCEANOGRAPHIC SCIENCES

BIDSTON

Routine storm surge forecasting using numerical models: procedures and computer programs for use on the CDC CYBER 205E at the British Meteorological Office

by

R. Proctor and R.A. Flather

I.O.S. Report No. 167

1983

ERRATA

P.23, line 10 should be "no $\bar{v}$-calculation"
P.25, line 15 read "A′,B′ and C′ (see.."
P.29, bottom, change figs 8,9 to figs 9,10
Pages 59 and 60 are reversed
P.162, line 13360 change +TAU(.. to -TAU(..

CONTENTS

1. Introduction

The development of numerical sea models for storm surge forecasting has been carried out at IOS Bidston for several years (Flather and Davies 1978, Flather 1981). Since 1978 one of these models has been run at the Meteorological Office to provide, on a routine basis, forecasts of storm surges for the Storm Tide Warning Service (STWS). The model runs twice a day throughout the "storm surge season" (September to April) and provides predictions up to 30 hours ahead. An integral part of the forecasting system is the use by the sea model of air pressure and surface winds predicted by an atmospheric numerical model at the Meteorological Office. The surge prediction scheme forms part of the routine Meteorological Office operational computer suite and runs shortly after the atmospheric model to take advantage of the most recent forecast.

At the beginning of 1982 the Meteorological Office replaced their main computer, an IBM 360/195, with a CDC CYBER 205E. Whereas the IBM machine was a serial processor type computer, the CDC machine is a vector processor. It was anticipated that the new machine would be an order of magnitude more powerful than its predecessor, due to the fundamental differences in its architecture. The increased computing power allowed new numerical models of the atmosphere to be introduced for operational weather forecasting.

These developments required a complete revision of the original surge prediction procedure and of the associated computer programs before the start of the 1982-83 season. The changes were required 1) because of the introduction of new atmospheric models, and 2) in order to make efficient use of the new computer.

The new procedure, based on the same dynamical equations as the original scheme, is fundamentally different in its organisation and operation. In the original scheme, developed on serial computers, the surge prediction was produced by carrying out a sequence of distinct operations, each of which covered the complete span of data required for the forecast. The basic steps were: 1) process and interpolate the meteorological data, 2) compute the tide, 3) compute the tide and meteorological effects together, 4) compute the storm surge by subtracting the results of 2) from those of 3) and output the required information. Each step involved reading data from one or more disc files, processing it, and writing the results in another disc file for use in a subsequent step. In the new scheme, the forecast proceeds hour-by-hour, and involves the same basic steps, but since the data to be passed from one step to the next now covers only one hour rather than the complete forecast period, it can be accomodated in memory thus obviating the need to write to and read from disc storage. A further gain in efficiency is achieved by carrying out input and/or output and calculations simultaneously.

The new design has resulted in a more robust scheme, that is, it has a greater capacity to recover a forecast in the event of machine failure. The design has also improved the computational efficiency which in turn has allowed the basic sea model, the continental shelf model (CSM) (Flather 1979), to be extended geographically to include areas of the Atlantic Ocean adjacent to the continental shelf; the new model being known as the continental shelf extended model (CSX).

The purpose of this report is to describe the models used in the surge forecasting procedure and to indicate their implementation on the Meteorological Office computer. As a consequence, a guide to the design of numerical sea models for use on a CYBER 205E is also given.

2. The forecasting procedure.

The storm surge forecasting procedure is based on dynamical numerical models of the atmosphere and of the sea. The atmospheric model is used to provide the essential forecasts of surface winds and air pressure which are then used in sea model calculations to give predictions of developing storm surges. The models are run twice per day, with forecast start times at 0000GMT and 1200GMT. Each forecast covers a 36 hour period but is not available from the computer until approximately 3 hours after the initial data time; thus the useful forecast produced is of length roughly 33 hours. The forecast is directed to the STWS, who use it and other information to issue warnings of abnormally high (or low) sea levels due to storm action to the regional Water Authorities and other responsible organisations. Forecasts are computed from September throughout the winter months until April. The procedure of operation is shown in the schematic, figure 1.

The Meteorological Office use two atmospheric models for routine weather prediction. One covers the northern hemisphere; the other limited area model provides higher resolution over the North Atlantic and western Europe. Both models have 15 levels in the vertical, unequally spaced in the σ-co-ordinate system employed, and use a staggered horizontal finite difference mesh. The pressure points of the limited area model, which provides data for the surge calculations, are shown in figure 2. With its horizontal latitude-longitude grid the spatial resolution is approximately 75 kilometres over the North Sea. Winds from the lowest level of the model (σ=0.997, level 1, height approximately 30 metres above sea level) and surface air pressures are extracted at hourly intervals. These are converted to wind stress and effective hydrostatic pressure respectively and then linearly interpolated to the sea model grid where the wind stresses and pressure gradients are applied. Two runs of the sea model are performed: in the first, the computation is for the tides only; in the second, the tides and the meteorological effects are computed together. The storm surge residuals are obtained by subtracting the results of the first run from those of the second. In this way surge-tide interaction, an extremely important process for surge propagation in shallow water, is accounted for.

For particular ports around the British Isles the forecast storm surge residuals are tabulated in the form required by the STWS, an example of which is shown in table 1. Computed tidal high (H) and low (L) water times are flagged for guidance, indicating the forecast surge residuals most likely to be of consequence for coastal flooding and navigation, respectively. At the STWS, these residuals are then added to the harmonically predicted astronomical tide to provide estimates of total levels. If these levels approach or exceed predetermined danger levels at a port, warning procedures are instigated. A description of STWS procedure can be found in Townsend, 1981.

As can be seen in figure 1, the initial condition for a surge forecast is taken from the previous forecast, normally at $t_s=12$. This means that errors from the earlier forecast are transferred into the current one. If the previous forecast was poor, the initial error may be substantial. One method of reducing the influence of a poor forecast is to precede each forecast by a hindcast surge calculation based on meteorological observations (Flather,1981). This method was examined by

Flather and Proctor (1983) for the case of the poorly predicted surge of New Years Day 1981 and found to improve the predictions. Subsequently, the scheme was adopted for the 1981-82 surge season. Of course, if the meteorological forecasts are accurate, there should be little improvement in the quality of the predictions as a result of incorporating hindcast information. For the season 1982-83 no hindcast data were available, hence the forecast mode only in figure 1. It is, however, relatively easy to include such data if and when they become available although the improved forecasts of the new atmospheric model may partially eliminate the need for such information.

3. The sea model.

Dynamical equations

The sea model is based on the two-dimensional depth averaged equations of continuity and motion written in spherical polar form

$$\frac{\partial \zeta}{\partial t} + \frac{1}{R\cos\phi}\left\{\frac{\partial}{\partial \chi}(D\bar{u}) + \frac{\partial}{\partial \phi}(D\bar{v}\cos\phi)\right\} = 0 \tag{1}$$

$$\frac{\partial \bar{u}}{\partial t} + \frac{\bar{u}}{R\cos\phi}\frac{\partial \bar{u}}{\partial \chi} + \frac{\bar{v}}{R\cos\phi}\frac{\partial}{\partial \phi}(\bar{u}\cos\phi) - 2\omega\sin\phi.\bar{v}$$
$$= \frac{g}{R\cos\phi}\frac{\partial \zeta}{\partial \chi} - \frac{1}{\rho R\cos\phi}\frac{\partial p_a}{\partial \chi} + \frac{1}{\rho D}(F_s - F_B) \tag{2}$$

$$\frac{\partial \bar{v}}{\partial t} + \frac{\bar{u}}{R\cos\phi}\frac{\partial \bar{v}}{\partial \chi} + \frac{\bar{v}}{R}\frac{\partial \bar{v}}{\partial \phi} + \bar{u}^2\tan\phi + 2\omega\sin\phi.\bar{u}$$
$$= \frac{g}{R}\frac{\partial \zeta}{\partial \phi} - \frac{1}{\rho R}\frac{\partial p_a}{\partial \phi} + \frac{1}{\rho D}(G_s - G_B) \tag{3}$$

where the notation is :

χ, ϕ	east-longitude and latitude respectively
t	time
ζ	elevation of the sea surface
$\bar{u},\bar{v}$	components of depth mean current, $\underset{\sim}{q}$
D	total depth of water (=h+ζ)
h	undisturbed water depth
R	radius of the Earth
ω	angular speed of rotation of the Earth
g	acceleration due to gravity
ρ	density of sea water, assumed uniform (=1025 kg/m^3)
p_a	surface atmospheric pressure
F_s ,G_s	components of wind stress $\underset{\sim}{\tau_s}$ on the sea surface
F_B ,G_B	components of bottom stress $\underset{\sim}{\tau_B}$

The quadratic law is used to relate the bottom stress to the depth mean current i.e.

$$\underset{\sim}{\tau_b} = k\rho\underset{\sim}{q}|\underset{\sim}{q}|$$

with k=0.0025

<u>Initial and boundary conditions</u>.

Equations (1) - (3) are to be solved starting from a prescribed initial state of elevation and motion, and subject to boundary conditions on land and open sea boundaries. The sea model integrations normally start from zero initial data at the beginning of the season, but once the sequence of forecasts is under way, the initial data for the current forecast is taken from the 12 hour fields (t_s=12 in figure 1) of the previous forecast. In the event of a forecast, or forecasts, being lost because of computer failure or some other cause, the 24 (t_s=24) and 36 (t_s=36) hour fields are available as backup initial conditions to allow the sequence to continue. If more than two successive forecasts are missed, a restart is performed with zero initial data.

At a coastline, the boundary condition is

$$q_n=0 \qquad (4)$$

where q_n is the component of depth mean current along the outward-directed normal to the boundary.

On an open sea boundary, a "radiation" condition is used

$$q_n=q_n^M + q_n^T + c/h(\zeta - \zeta^M - \zeta^T) \qquad (5)$$

where $c=(gh)^{1/2}$ and q_n^M, ζ^M and q_n^T, ζ^T describe the input of meteorological (M) and tidal (T) origin. The condition (5) seeks to prevent the artificial reflection from the open boundary of disturbances generated within the model by making them propagate out, locally, as free progressive waves. The tidal input q_n^T, ζ^T is expressed in standard harmonic form e.g.

$$\zeta^T = \sum_{i=1}^{n} f_i H_i \cos(\sigma_i t + u_i + v_i - G_i)$$

where H_i and G_i denote the harmonic constants, amplitude and phase, for the ith constituent, σ_i the speed, v_i the phase of the corresponding equilibrium constituent at Greenwich at time t=0 (the start of a forecast) and f_i, u_i are nodal factors allowing for the 18.6 year variation in amplitude and phase of the constituent. The values of v_i, f_i and u_i are computed at the start of each forecast using orbital elements s, χ and n thus:

$$v_i = I_{1,i} s + I_{2,i} \chi + \sigma_i t$$

$$f_i = a_{1,i} + a_{2,i} \cos(n) + a_{3,i} \cos(2n)$$

$$u_i = b_{1,i} \sin(n)$$

where coefficients I_1, I_2, a_1, a_2, a_3 and b_1 are taken from Doodson (1921). The tidal input is derived from offshore measurements (Cartwright et al 1980) and model experiments (Flather, 1980). Only M_2 and S_2 constituents are included. The surge input q_n^M, ζ^M is assumed to be

$$q_n^M = 0, \quad \zeta^M = (\bar{p}_a - p_a)/\rho g$$

where $\bar{p}_a$ is 1012mb.

The equations (1) - (3) are solved by means of finite difference techniques (see Appendix I) on the grid shown in figure 3. The grid spacing is 20′ in latitude and 30′ in longitude and the timestep of integration is 144 seconds. This mesh covers the north west European Continental Shelf and adjacent Atlantic waters. Deep areas of the neighbouring ocean would necessitate a substantial reduction in the timestep and are consequently excluded.

The meteorological forcing terms in (2) and (3) are computed from surface pressure, p_a, and wind, $\underset{\sim}{W}$, defined hourly at grid points of the atmospheric model. This model has a staggered grid for pressures and winds: pressure points are marked (X) in figure 3. On the atmospheric model grid, the pressures are converted to equivalent hydrostatic elevations and the winds are converted to wind stresses using a quadratic law

$$\underset{\sim}{\tau_s} = c_D \rho_a \underset{\sim}{W} |\underset{\sim}{W}|$$

where ρ_a, the density of air is assumed to be 1.25 kg/m^3

and

$$c_D * 10^3 = A0 + A1W \qquad \text{with W in m/s}$$

where A0=0.63 and A1=0.066 (Smith and Banke, 1975).

The data are then linearly interpolated to the sea model grid and the hydrostatic elevations are used to compute pressure gradients, using centred differences, and surge input on the open sea boundary.

4. The CDC CYBER 205E

The Meteorological Office obtained a CYBER 205E vector processor mainly to pursue research into possible techniques for forecasting fluctuations in the Earth's climate. It was expected that this machine would give an order of magnitude increase in power over that of its predecessor, an IBM 360/195. In table 2 are detailed some of the design and installation features of the CYBER. The table refers to the present status of the installation, which can be upgraded by doubling the number of pipes, quadrupling the memory and adding more peripherals. At present, no magnetic tape facilities are available directly on the CYBER. Data can, however, be passed through a link to another computer at the Meteorological Office, an IBM 370/158, and then stored on magnetic tape. The IBM machine acts, among other things, as a front line processor for the CYBER.

a) Programming considerations.

(i) In its normal mode of operation, the CYBER uses 64-bit arithmetic. The use of 32-bit arithmetic instead of 64-bit arithmetic can increase the computational speed of the CYBER by a factor of two. At the time of writing the storm surge suite no 32-bit Fortran compiler was available. The programs were, therefore, written to use 64-bit precision. Even without such a compiler, 32-bit arithmetic can be performed by the generation of 32-bit vector instructions using CYBER 200 Fortran. This allows machine instructions to be inserted into a program by means of CALL statements to subroutines with special names. Subsequently, a 32-bit Fortran compiler has been introduced which reduces the need for such CALL's.

(ii) Efficiency is also enhanced by the use of triadic statements. Examples of such statements are as follows

$$A = BX + C$$

$$A = X(B + C)$$

where A,B,C are vectors and X is a scalar.
Such combinations on the CYBER are executed as single instructions. More complex statements than these should, where possible, be broken down into triads.

(iii) Vector length. Vectors should be made as long as possible, up to the maximum of 65535. This is because of the way instructions are performed. For each instruction, data is streamed through the vector pipes from the memory and then back to memory. The loading of the first pair of data points into the two pipes is subject to a start-up time. For short length vectors this time can be a substantial proportion of the total time taken to process the instruction. For example, to perform a multiplication in 32-bit arithmetic on a vector length of 200 is 50% efficient (due to start-up time) whereas it is 95% efficient on a vector length of 10000. Here, efficiency is measured by the amount of time taken to perform the calculation compared to that taken by the start-up time plus the calculation.

Rationalisation of the calculations so that operations are performed on long vectors of contiguous data should be carried out. If two dimensional arrays are used in a program they should be treated as one-dimensional arrays, i.e. two-dimensional arrays A(I,J) are stored columnwise so calculations should proceed column by column, not row by row.

Dickinson (1982) cites an example of such rationalisation in the transfer of a 10-level atmospheric model from a serial processor machine to a vector processor. On the serial machine, calculations were performed through the 10 levels, grid point by grid point. This meant the maximum vector length was 10. On the vector machine, the problem was turned around and the calculations performed grid point by grid point for each of the 10 levels. Thus a vector length of 20000, the total number of grid points in the model, could be constructed.

b) Programming features.

To achieve a high performance on the CYBER, use has to be made of CYBER 200 Fortran. The notation is neat and concise and if the program is straightforward, simple to follow. Examples of CYBER 200 Fortran will be taken from the surge programs to illustrate various features.

(i) DO-Loops

Consider the DO-loop

```
          DO 100 I=1,ITOT
          ZZ(I)=0.0
      100 CONTINUE
```

there are two ways this could be written in CYBER 200 Fortran. Either

```
          ZZ(1;ITOT)=0.0      (line 2930 in program GESMOD)
```

the general form of which is

```
          ZZ(IS;ILEN)=0.0
```

where IS is the start address and ILEN is the length of the vector, thus

```
          ZZ(10;100)=0.0
```

is the same as

```
          DO 300 I=10,109
          ZZ(I)=0.0
      300 CONTINUE
```

or, by using a DESCRIPTOR thus:-

```
          DESCRIPTOR ZD
          ASSIGN ZD,ZZ(1;ITOT)
          ZD=0.0
```

Here ZD is a variable name which, when assigned to a vector, points to the storage locations set up in the ASSIGN statement (as in subroutine PARFO, lines 8020,8130 and 8140). Although it takes three lines to perform the operation using a descriptor, calculations involving array ZZ, elements 1 to ITOT, can be performed elsewhere in the program just by using the descriptor ZD. A descriptor can also be reassigned to another array of locations or to a different sequence of locations in the original array at any time in the program, thus

```
          ASSIGN ZD,UU(1;ITOT)
```

and ZD now points to array UU, locations 1 to ITOT.

(ii) Use of BIT control vectors.

By constructing a vector of BITS, where each BIT has the value '0' or '1', operations can be performed on selected parts of real or integer vectors. BIT vectors can be used to perform indirect addressing through the use of special CALL functions. In the storm surge model, BIT vectors (or BIT masks) are used to control the storage of calculations of elevation and depth mean currents, to extract the required meteorological data from the atmospheric model arrays, to allocate open sea boundary points, to compress arrays for output to disc and many other similar operations.

Construction of a bit mask.

As an example, consider the construction of the bit mask BITZ which identifies open sea elevation points at which the continuity equation (1) is to be solved. The array has to be dimensioned and declared as type BIT (lines 220 and 170 in program SETUP). i.e.

```
DIMENSION BITZ(3072)
BIT BITZ
```

Then a descriptor (also of type BIT) is assigned to the array

```
DESCRIPTOR BITZD
BIT BITZD
ASSIGN BITZD,BITZ(1;ITOT)
```

The mask is established by setting the individual bits in BITZ to have value '1' at an elevation point i.e. where the 'label' of the point is greater than or equal to 100 (see next section for a complete definition of the labels). A descriptor, LABD, is assigned to the array of labels, LAB,

```
DESCRIPTOR LABD
ASSIGN LABD,LAB(1;ITOT)
```

then the mask is created thus

```
BITZD=LABD.GE.100
```

(lines 3000 to 3020 in SETUP).

(iii) Special subroutines.

CYBER 200 Fortran includes special subroutines which are used to place machine instructions in the object code. The main uses made of these special functions by the sea model are :

a) the concurrent I/O of data,

```
          CALL Q7BUFIN(NRIT,ZIO,9,'SMALL') line 3030 GESMOD
          CALL Q7BUFOUT(NWFT,ZIO,9,'SMALL')      6080 GESMOD
```

thus allowing data to be streamed into or out of a specified area of memory while some other operations are being performed. Arrays which are used in these statements must be declared in named COMMON blocks, e.g.

```
          COMMON/BUFIO/ZIO(1536,3)            line 380 GESMOD
```

This is because the arrays must be aligned on the boundary of a unit of memory, a SMALL page. If this is not achieved, the program will fail. Named COMMON blocks can be aligned in memory by use of the GRSP parameter on the LOAD instruction (see later).

b) the removal of unwanted elements from an array using Q8MASKV. This routine has the format

```
          CALL Q8MASKV(G,,A,,B,Z,C)
```

where G is an 8-bit designator (G-bit) which allows alternative operations to be performed on the arrays specified, Z describes a BIT mask such that if

	$Z_n=1$	element	$A_n \rightarrow C_n$
and	$Z_n=0$	element	$B_n \rightarrow C_n$

e.g.

```
          DESCRIPTOR ISEAD,LABD,BITZD
```

```
ASSIGN ISEAD,ID(1;ITOT)
ASSIGN LABD,LAB(1;ITOT)
ASSIGN BITZD,BITZ(1;ITOT)
ISEAD=1
LABD=0
CALL Q8MASKV(X'00',,ISEAD,,LABD,BITZD,ISEAD)
```

thus each element of ID is set to 0 or 1 depending on whether the corresponding element of BITZ is '0' or '1'. Here the G-bit performs no additional operation (for an explanation of G-bits see Appendix II). There are many such CALL Q8... routines for performing a variety of operations e.g. to add or subtract vectors, multiply or divide, perform logical operations or merge vectors.

Additional functions used in the model are to compress, Q8VCMPRS, and expand, Q8VXPND, arrays for I/O, thus saving disc space. The statement

```
ZIO(1,1;IINZ)=Q8VCMPRS(ZT(1;ITOT),BITZ(1;ITOT);ZIO(1,1;IINZ))
```

(line 6000, GESMOD) compresses the array ZT by filling the first IINZ elements of ZIO with elements of ZT at which the corresponding element of BITZ has a value '1'. This is equivalent to the following coding

```
      K=0
      DO 1 I=1,ITOT
      J=ID(I)
      IF(J.EQ.0) GOTO 1
      K=K+1
      ZIO(K,1)=ZT(I)
    1 CONTINUE
```

where each element of ID has previously been set to either 0 or 1 depending on the bit values of BITZ. IINZ is the number of '1' bits in BITZ. The function Q8VXPND operates in the reverse manner.

All special functions and subroutine calls are described in detail in the CYBER 200 Fortran Reference Manual (1981).

c) Data storage.

Data (and programs) on the CYBER are stored in PAGES. PAGES can be either LARGE or SMALL. A SMALL PAGE is 4 blocks of 512 64-bit words (i.e. 2048 words) and a LARGE PAGE is 128 blocks. Prior to storing data sets, sufficient space should be REQUESTed on a disc in units of small pages (in practice, as the number of 512 word blocks).

When using concurrent I/O i.e. Q7BUFIN, Q7BUFOUT, the size of the array to be moved is given in the argument list (see above) so the size of any arrays used in I/O must be known in advance of the CALL. The concurrent I/O of BITS is performed in the same way, the units are still PAGES (thus one small page corresponds to 2048*64 = 131072 bits).

When the size of arrays to be input is variable, a header of one block in which the number of pages occupied by the data stream is specified, can be input prior to the main data stream.

5. Computing philosophy.

The basic intention has been to make the sea model as general as possible so that, given a set of input data defining a particular sea area, parameters are generated which directly control the sea model computations for that area. The generalisation extends to the peripheral programs, for example, the generation of tidal input and the meteorological interpolation parameters. Figure 4 illustrates the main inputs to the sea model. The input data required to define the sea model consists of : the number of rows and number of columns in the matrix, the sizes of the mesh sides of the elements, the latitude origin of the matrix (centre of the most north-westerly element), the undisturbed water depth at the centre of each element and a matrix of labels defining the operations to be carried out for each element. The label matrix constitutes an important part of the initialisation procedure because, from it, the calculation areas are resolved and the open boundary locations determined. Each label consists of an integer, generally of three digits. The first digit refers to the elevation point, the second to the u-point, and the third to the v-point, within the associated grid element: a value 1 indicating that the appropriate equation must be solved at the point; a value zero indicating that no operation is required there as, for example, when a point is on the land. A value 2, rather than 1, is used to indicate that the normal calculation at the point must be modified because of the presence of an open sea boundary. This classification leads to the following labels, illustrated in Figure 5.

111 - open sea point with ζ, $\bar{u}$ and $\bar{v}$ calculation

110 - open sea point, ζ, $\bar{u}$ calculation, closed $\bar{v}$-boundary

101 - open sea point, ζ, $\bar{v}$ calculation, closed $\bar{u}$-boundary

100 - ζ calculation, closed $\bar{u}$ and $\bar{v}$ boundaries

222 - open boundary point; omit advection from $\bar{u}$ and $\bar{v}$ calculations

221 - open boundary point; omit advection from $\bar{u}$ calculation normal $\bar{v}$ calculation

220 - open boundary point; omit advection from $\bar{u}$ calculation, closed $\bar{v}$-boundary

212 - open boundary point; omit advection from $\bar{v}$ calculation normal $\bar{u}$ calculation

211 - open boundary point, normal $\bar{u}$ and $\bar{v}$ calculation

210 - open boundary point, normal $\bar{u}$ calculation, no $\bar{v}$ calculation

202 - open boundary point, omit advection from $\bar{v}$ calculation, no $\bar{u}$ calculation

201 - open boundary point, no $\bar{u}$ calculation, normal $\bar{v}$ calculation

200 - open boundary point, no $\bar{u}$ or $\bar{v}$ calculation

-1 - element exterior to, but adjacent to, an open boundary point - no calculation required

0 - null calculation point

In Figure 5 • represents a ζ-point, + a $\bar{u}$-point and × a $\bar{v}$-point. Circled symbols have prescribed boundary values. At open boundary points, advection is excluded from the equations of motion whenever the computation of a spatial gradient requires a value which is outside the computation area e.g. for a point with label 221 or 220, the gradient $\partial \bar{u}/\partial \phi$ cannot be evaluated.

Bit masks control the actual sea model calculations. Each bit mask is an array of single bits covering the model area with the bit set to either '1' or '0' depending on whether or not a calculation is to take place there. Calculations of ζ ,$\tilde{u}$ and $\tilde{v}$ are performed everywhere in the matrix but their values are stored only at locations where the bit value is '1'. Bit masks are also used to derive open boundary input, to compress arrays for output and to control the interpolation of meteorological data.

Most of the computational effort in the forecasting suite is in the numerical model integration. To make this efficient means taking advantage of the features of the CYBER. One of these features is the facility for concurrent I/O. This means that data can be input or output simultaneously with some other operation e.g. a calculation. The model is written to take advantage of this feature. Instead of performing a full forecast for tide, then a forecast for tide plus surge and then computing the residuals, as performed on a serial computer, the model is arranged to calculate tide, tide plus surge and residuals on an hourly basis. A schematic of the model computation procedure is shown in figure 6. All the necessary control and model data is input at the beginning of the program. Calculations are started for a number of hours, N. At the beginning of an hour, meteorological data, if required, is input and at the same time, tidal computations over the model area are carried out (40 iterations using a timestep of 144 seconds for CSX). Once the tidal computations are complete, a check is made to ensure completion of the input of meteorological data. Then, the final values of the tidal computations for that hour are output if necessary and, simultaneously, the processing of the meteorological data is carried out. After checking that output of the tidal data is complete, tide and surge are calculated for that hour. The final values of the hour's computation of tide and surge are then output if required while the table of elevations is updated. The output of tide plus surge is then checked for completion, followed by the output of residual arrays which continues while any printing of data is performed. Finally, a check is made on the completion of residual output before the cycle begins again for the next and successive hours.

A further economy was made by rearranging the model so that there were fewer land (i.e. null) points in the mesh. Patches of sea were moved to fill appropriately sized land areas, allowance being made for overlap points. Thus, model CSX has 52 rows and 57 columns, 2964 elements of which 1340 are sea points. By rearranging the grid and moving areas A,B and C to their new locations A,B and C (see figure 7) the new array has 46 rows, 44 columns and 2024 elements in all still with 1340 internal points. Efficiency is improved since calculations are carried out at fewer points overall, and because the number of points at which the result is discarded by the masking process is also reduced. The improvement far outweighs the additional data transfers required to provide for calculations adjacent to the patch boundaries.

At present, all computations are carried out using 64-bit arithmetic. This accuracy is unnecessary for the type of computations performed (Dickinson,1982) and 32-bit precision would suffice. At the time of writing the programs no half precision compiler instruction was available which meant the coding had to be performed entirely using the special CALL instructions, and this was considered impracticable in the length of time available. However, the necessary compiler option is now available and its use could increase the efficiency without any loss in numerical accuracy. This step may be required in the future to partially offset the increase in running costs for higher resolution sea models.

6. Computer programs for storm surge forecasting.

Four computer programs constitute the surge forecasting suite:

SETUP - this routine takes the basic sea model data and computes the necessary bit masks, storing them in a file using concurrent I/O.

SETUPT - takes the open sea boundary tidal input constituents in harmonic form and stores then in a file.

METSET - sets up the necessary parameters and bit masks for the processing of the meteorological data.

GESMOD - the sea model program; processes met. data, computes tide, tide plus surge and residuals and outputs the results.

Flow charts for these four programs are given in figures 8,11,12 and and 13 respectively. The program listings and input data sets are given in Appendix III.

Program SETUP.

Given the basic input of a model i.e. the number of rows, number of columns, grid size, latitude co-ordinate of origin, matrix of labels and matrix of depths, the program computes latitude dependent terms and bit masks for controlling ζ ,$\bar{u}$,$\bar{v}$ calculations and open boundary allocations. Allowance is made for the extraction of a subset of the matrix and the relocation of parts (patches) of the matrix.

Input data.

NRX,NCX	- number of rows and columns in matrix
NRR,NCC	- number of rows and columns in final output matrix

JRUN,ILEG	- run and leg numbers for run identification
ITIT	- array for secondary title, used in STWS output format
IZET	- array of point addresses for entry in the tables (maximum of 60)
ITIP	- array of names identifying points in IZET
LMI	- frequency, in hours, of meteorological input
A0,A1	- wind stress parameters, assuming a linear variation of drag coefficient with wind speed

Sea model data, calculated in SETUP is input from file CPSMD3. Meteorological interpolation data is input from file CSPID1. Arrays of variables in compressed form for initial and final conditions of tide and tide plus surge are found in files CPTID1,CPTID2,CPSUR1,CPSUR2. Winds and pressures at hourly intervals covering a 36 hour period can be input from files WINDFMF and FMFCSP respectively. The model, being a generalised one, has the facility to allow prescribed residual boundary input, read from logical unit 20. In the surge forecasting model CSX this is not used and boundary residual input is derived from the hydrostatic elevation. Boundary tidal data (for up to 4 constituents) is input from file CSPBIT.

Output data

The usual output of the surge model during operational running is the table of surge elevations at standard ports, table 1. This table is

IZOBN,IZOBS,IZOBW,IZOBE	- arrays of addresses of north, south, west and east located open boundary points
NRR,NCC	- numbers of rows and columns
ITOT	- number of elements in the matrix (=NRRxNCC)
IINZ,IMEU,IMEV	- numbers of ζ ,$\bar{u}$ and $\bar{v}$ internal sea points
IINUX,IINVX	- numbers of $\bar{u}$ and $\bar{v}$ internal sea points including open boundary points
IOBZ,IOBU,IOBV	- numbers of ζ ,$\bar{u}$ and $\bar{v}$ open boundary sea points
ICN,ICS,ICW,ICE	- numbers of north, south, west and east open sea boundary points
DX,DY	- longitude and latitude grid size

The array BITIO contains:

BITZ	- bit mask of internal ζ points
BITU	- bit mask of internal $\bar{u}$ points
BITV	- bit mask of internal $\bar{v}$ points
BITUX	- bit mask of internal and open boundary $\bar{u}$ points
BITVX	- bit mask of internal and open boundary $\bar{v}$ points
BITZO	- bit mask of open boundary ζ points

This data is stored in a file: CPSMD3.

For the surge forecasting model shown in figure 3, the bitmask of internal ζ points is displayed in figure 8; the bit mask after relocation of patches is shown in figure 9.

Program SETUPT

Allowance is made for up to four tidal constituents to be input to the model - this may be increased in the near future. For ζ , $\bar{u}$ and $\bar{v}$, harmonic constants H_i and G_i for NCON constituents are stored in the form $H_i\cos(G_i)$ and $H_i\sin(G_i)$.

Input data

IOBZ,IOBU,IOBV	- number of open boundary ζ ,$\bar{u}$ and $\bar{v}$ points
NCON	- number of tidal constituents,

then for each constituent i

SIG	- speed of the constituent, i, in degrees/hour
ATC	- coefficients for nodal factor, f_i
BTC	- coefficients for nodal factor, u_i
ICTC	- coefficients for v_i , the phase of the equilibrium tide at Greenwich at forecast start time, t=0
Z1,Z2	- $H_i\cos(G_i)$ and $H_i\sin(G_i)$ of ζ along the open boundary
U1,U2	- $H_i\cos(G_i)$ and $H_i\sin(G_i)$ of $\bar{u}$ along the open boundary
V1,V2	- $H_i\cos(G_i)$ and $H_i\sin(G_i)$ of $\bar{v}$ along the open boundary

Output data

NCON

then for each constituent

SIG, ATC, BTC, ICTC,

Z1,Z2

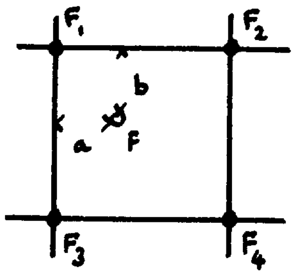
U1,U2

V1,V2

This data is stored in a file: CSPBIT in unformatted form.

Program METSET

This program computes all the necessary parameters for the linear interpolation of surface winds and air pressures from the atmospheric model grid onto the storm surge model grid. Allowance can be made for the relocation of patches. The necessary subset of winds and pressures is determined by the construction of bit masks and coefficients, a and b, are computed for the linear interpolation of each parameter i.e.



● atmospheric model point

o sea model point

such that $F = (1-b)(F_1(1-a)+F_2 a) + b(F_3(1-a)+F_4 a)$

Input data

NRX,NCX	- numbers of rows and columns in the sea model
CHIN,PHIN	- longitude and latitude of the most north-westerly elevation point
DX,DY	- longitude and latitude grid increments of the sea model

NRR,NCC,IFX,JFX,NPATCH - numbers of rows and columns in new matrix, start row and column numbers of old matrix for the new matrix, and number of patches to be relocated

$*\left\{\begin{array}{ll}\text{IXST,JXST,NCT,NRT,IF,JF} & \text{- patch relocation data} \\ \text{LABX} & \text{- matrix of labels}\end{array}\right.$

* omit if NPATCH = 0 and set NRR=NRX, NCC=NCX

In addition, the header information from the wind (WINDFMF) and pressure (FMFCSP) data files are input using concurrent I/O. This data defines the wind and pressure points in the atmospheric model.

Output data

Two arrays DAM and BITIM are output using concurrent I/O. DAM contains, via an EQUIVALENCE statement, all the REAL and INTEGER arrays necessary for the meteorological interpolation. BITIM contains, via an EQUIVALENCE statement, all the BIT arrays to control the wind and pressure extraction.

The array DAM contains:

IIMU,IIMV,IIMZ - one-dimensional arrays of atmospheric model point numbers at sea model points for the $\bar{u}$ and $\bar{v}$ components of wind and the air pressures

AUX,BUX - interpolation coefficients at sea model points for the $\bar{u}$ component of wind

AVX,BVX - interpolation coefficients at sea model

	points for the $\vec{v}$ component of wind
AZX,BZX	- interpolation coefficients at sea model points for the air pressures
IDM	- array of model constants, where
IDM(1)	- number of wind elements required to perform the interpolation
IDM(2)	- number of wind elements in the atmospheric model
IDM(3)	- array size, in small pages, of the wind data
IDM(4),IDM(5)	- number of columns and rows of the matrix of winds required from the atmospheric model
IDM(6)	- number of pressure elements required to perform the interpolation
IDM(7)	- number of pressure elements in the atmospheric model
IDM(8)	- array size, in small pages, of the pressures
IDM(9),IDM(10)	- number of columns and rows of the matrix of pressures required from the atmospheric model

The array BITIM contains:

BITW	- bit mask for control of wind data extraction
BITP	- bit mask for control of pressure data extraction

This data is stored in file: CSPID1.

Subroutines called are ARPRIN to print out the arrays of either integer point numbers or interpolation coefficients.

Arguments are:

A	- real array
IA	- integer array
NC,NR	- number of columns and rows in matrix
IFO	- indicator for real or integer array

Program GESMOD

The sea model program. Given a set of control parameters the model will compute the tide, introduce meteorological data, compute the tide plus surge, evaluate surge residuals and output the results. Use is made of the model data computed in SETUP and the meteorological control data computed in METSET. Output from the model can consist of arrays of residual elevation and depth mean currents at regular intervals in time, tables of surge elevation at specific points through time in STWS format (see table 1) and arrays of tide and tide plus surge at specified intervals for use as initial conditions for a following forecast.

Input data

From a file CPCF1 logical unit numbers defining the I/O of tide and tide plus surge arrays.

NRIT, NRIS	- input devices for tide and tide plus surge
NWFT,NWFS	- output devices for tide and tide plus surge

From a file CPCDS basic control parameters are input:

IZZ	- array for a general title
DT,FR	- timestep (in seconds) and bottom friction coefficient

DCRIT,ZCRIT	- parameters determining the wetting and drying of elements in shallow water regions (for details of the wetting and drying procedure see Flather and Heaps, 1975)
ITS	- determines the calculation of tide, or surge or tide plus surge
LHR	- number of hours of integration
LPR	- frequency, in hours, of array printout
LET	- frequency, in hours, of entries in tables
LPUN	- control parameter for storage of tables
IPCL	- printout control parameter
LGDG	- frequency, in hours, of storage of arrays
LGDGE	- last time for storage of arrays
ILAG	- time lag, in hours, between initial data and meteorological data
ISWSC	- control parameter for input of boundary surge currents
IPCW	- control parameter for printout of meteorological data
NPRTS	- number of points for entry in the tables
IPCP	- controls printout of variable; tide, tide plus surge or residual
NWFR	- logical unit number/ control parameter for storage of residual arrays
IROB	- control parameter for open boundary surge input

JRUN,ILEG	- run and leg numbers for run identification
ITIT	- array for secondary title, used in STWS output format
IZET	- array of point addresses for entry in the tables (maximum of 60)
ITIP	- array of names identifying points in IZET
LMI	- frequency, in hours, of meteorological input
A0,A1	- wind stress parameters, assuming a linear variation of drag coefficient with wind speed

Sea model data, calculated in SETUP is input from file CPSMD3. Meteorological interpolation data is input from file CSPID1. Arrays of variables in compressed form for initial and final conditions of tide and tide plus surge are found in files CPTID1,CPTID2,CPSUR1,CPSUR2. Winds and pressures at hourly intervals covering a 36 hour period can be input from files WINDFMF and FMFCSP respectively. The model, being a generalised one, has the facility to allow prescribed residual boundary input, read from logical unit 20. In the surge forecasting model CSX this is not used and boundary residual input is derived from the hydrostatic elevation. Boundary tidal data (for up to 4 constituents) is input from file CSPBIT.

Output data

The usual output of the surge model during operational running is the table of surge elevations at standard ports, table 1. This table is printed at the Met. Office for STWS use and is also stored in a file CPTAB1. A copy of this file is transferred through the link to the IBM 370/158 and appended to an archive file which has the capacity to hold 10 days of forecasts, thus providing the facility to examine any interesting surge events in retrospect. Tables of tide are also stored in the same file and are used to provide an indicator of high and low waters. The archive file needs to be cleared at regular intervals to prevent the allocated disc space being exhausted. Arrays of variables are output at 12 hourly (or other) intervals into files CPTID1 or CPTID2 for the tide and CPSUR1 or CPSUR2 for the tide plus surge for use as initial conditions for the next forecast. The use of CPTID1 or CPTID2 and CPSUR1 or CPSUR2 alternates with each successive forecast. Residual arrays of elevation and depth mean currents may be output at hourly intervals to file CSPRES. These might be used to obtain an overview of a particular surge development, to study a particular area in detail or to provide boundary input data to any other limited area model which is enclosed by the surge model e.g. of the southern North Sea.

Eight subroutines are called by the main program:

(1) VDAY - input arguments IDAY, IMNTH, IYR
return arguments IVDY

Given the day, month and year, the day number in that year is returned.

(2) DATEB - input arguments IVDY,IYR
return arguments IDAY, IMNTH, IYEAR

Given the day number and the year, the day, month and year are returned.

(3) STARTI - input arguments NW, JTIM
return arguments IHLAP

Given the start times of wind, pressure, tide and tide plus surge data, JTIM, the number of hours elapsed since 0000GMT 1/1/1982 is computed for each dataset and returned, IHLAP. This information is used to select the correct initial data for the current forecast. NW is the logical unit number for printed output.

(4) METPROC - input arguments ITOT, A0, A1, ROA, RO, GC,

PMEAN, LOOP, IPCW

return arguments, none

Given the array size, ITOT, windstress drag coefficients, A0 and A1, the densities of air, ROA, and of water, RO, the gravitational constant, GC, a reference pressure, PMEAN the LOOP counter and the printout control parameter IPCW, winds and pressures are extracted from the atmospheric model arrays, converted to stress and equivalent hydrostatic elevation and then linearly interpolated on to the sea model mesh.

This subroutine calls two other subroutines

(i) LIMP - input arguments IIMU, AUX, BUX, ISO, NCCMS, UO

Linear interpolation of atmospheric model variable, UO, between points IIMU using coefficients AUX and BUX. ISO is the number of points in the sea model, NCCMS is the number of columns in the atmospheric model. Interpolated values at sea model points are returned in UO.

(ii) PRIW - input arguments NW, IS,IE, JS,JE, UO, VO, NCOMP

return arguments, none

Print either an array of pressures UO (NCOMP=1), or two arrays of winds, UO, VO (NCOMP=2) where IS, IE, JS, JE denote start and end values of columns and rows respectively. NW is the printout logical unit number.

(5) PARFO - input arguments ITOT

return arguments, none

Given the number of elements in the model, ITOT, and a control parameter, IPCL, found in COMMON, arrays of either ζ (IPCL=10), $\bar{u}$ and $\bar{v}$ ($0<\text{IPCL}<10$) or ζ ,$\bar{u}$ and $\bar{v}$ ($\text{IPCL}>10$) are prepared for printing. Land areas are allocated a large number and so appear as asterisks in the final printout. This subroutine calls one subroutine:

PRNTZ - input argument A

return arguments, none

This routine prints an array, A, picking up necessary parameters from COMMON.

(6) SHPN - input arguments IYR, IVDY

return arguments SSS, HHH, PPP, CN, PPPP

Given the year, IYR, and day number, IVDY, this routine calculates the orbital elements s(SSS), h (HHH), p(PPP), n(CN) and p_1 (PPPP) describing the declinations of the sun and moon at 0000GMT/IVDY/IYR. For further detailed information concerning these parameters see Doodson and Warburg (1941), Chapter 7.

(7) GESMEQ - no arguments

This routine carries out integrations of the equations of continuity and depth mean motion for a number of timesteps, NTPH (found in COMMON).

(8) PELTA - input arguments JET, NPRTS, ITIT, ITIP, IHS,

IDAY, IMNTH, IYR, IPCP

return arguments, none

Given the length of time series, JET and the number of ports in the table, NPRTS, this routine prints out a table in STWS format for a forecast JET hours in length for NPRTS ports, starting at time IHS,IDAY,IMNTH,IYR. ITIT is a general title, ITIP is an array of abbreviated port names for identification. IPCP is a control parameter which permits either tables of residuals only (IPCP=0) or tables of tide plus surge, tide and residuals (IPCP$\neq$0).

7. Model implementation on the CYBER

In the following eight subsections details will be given of how to run programs on the CYBER - from how to gain access to the machine up to running the surge forecasting suite. The subsections can be classified as follows:

a) Access to the CYBER

b) Running simple jobs

c) File allocation

d) The use of UPDATE files

e) The use of CONTROLLEE files

f) Data files

g) Loading the surge forecasting suite

h) Running the surge forecasting suite

Sample job control language (JCL) will be given to illustrate these points.

a) Access to the CYBER

There is no direct access to the CYBER itself. All jobs must be submitted to the CYBER via the IBM 370/158. This means that a two-stage job is necessary: the first step to run on the IBM machine which will transfer the job across to the CYBER and the second job to run on the CYBER itself. Access to the IBM machine can be either from batch (at the Met. Office), from TSO (timesharing) or from RJE (remote job entry - workstation). We shall only consider RJE, the method used at Bidston.

A sample deck to run a job on the CYBER via the IBM machine is:

```
//D65IOSAB JOB (EXT,RF,XC1#1),IOS-OUT-JOB,PRTY=10
//*MAIN SYSTEM=P58
// EXEC CYBERSUB
  SUBMIT MF=C20,FILE=JCL,SYSOUT=A,DEST=TERM4PR1
//JCL DD *
blank card
   ∫
CYBER job
   ∫
//
```

The jobname, D65IOSAB, will be the same on the IBM and the CYBER. The output from the CYBER job will be routed back through the IBM to Bidston (TERM4PR1). If any other destination is required DEST= must be redefined.

<u>b</u>) <u>Running simple jobs</u>

The simplest job to run is a straightforward program compilation, load and execution thus:

```
USER(U=123456,AC=A1234)
RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.
FORTRAN,I=COMPILE,B=BIN1,O=B.
LOAD,BIN1,CN=GO1.
GO1.
7/8/9
   ∫
program
   ∫
```

```
     ≀
    7/8/9
     ≀
    data (if any)
     ≀
    7/8/9
    6/7/8/9
```

The USER card states the account information. The RESOURCE card specifies the resources the job requires. The FORTRAN card generates an input file, COMPILE, which contains the source program, compiles the program and writes the compiler-generated object module into file BIN1 using option B. The LOAD card takes the object module, BIN1, and generates a file, GO1, in which is stored the executable code. The card GO1 calls the file of the same name and executes the code. GO1 is known as a controllee file.

As can be seen, program structure is similar to that of the CRAY and in general terms can be considered as

```
     ≀
    CONTROL CARDS
     ≀
    7/8/9
     ≀
    PROGRAM
     ≀
    7/8/9
     ≀
    DATA
     ≀
    7/8/9
    6/7/8/9
```

the statements 7/8/9 and 6/7/8/9 are end of section and end of job cards and are multipunch characters formed by overpunching numbers 7,8 and 9 or 6,7,8 and 9. All C"BER control statements can be found in the CYBER 200 Operating System Manual (1980). User numbers (U=...) and account codes (AC=...) are obtained from the Met. Office.

If in the program, concurrent I/O is used, the COMMON blocks which hold the arrays for I/O must be declared on the LOAD card using the GRSP parameter (one for each COMMON block) e.g.

```
        LOAD,BIN1,CN=G01,GRSP=*BUFIO.
```

(see section 4(b).(iii)).

c) File allocation

If a program requires to read or write to data files, there are two ways of defining the files. Before describing these ways it is useful to understand how files are stored on the CYBER. To each USER number an allocation of disc space is made. This space is subdivided into two categories; general space and POOL space. The POOL space is really a subcatalog area of the general space and several POOL's may exist. Files can be stored either in the general space or in a POOL. Storing files in POOL's is encouraged as it makes data management easier.

Assume data file A already exists (file creation is discussed in subsection f)) and is resident in a POOL named SURGE. Before accessing the file the pool must be attached to the job thus

```
        PATTACH,SURGE.
```

This command must precede any request for any file in this pool. If a file (say B) is in the general space it is known as a PERMANENT file. Before accessing a permanent file it must be attached to the program by the command

```
ATTACH,B.
```

Files can be called by the program either 1) from within the program or 2) at execution of the program. 1) If the files are called from within the program, they are defined on the PROGRAM declarator card which is the first card in the source deck e.g.

```
PROGRAM TEST(UNIT5=INPUT,UNIT6=OUTPUT,UNIT10[,,4]=A)
```

Here the logical unit 5 is assigned to the card input stream, unit 6 is assigned to line printer output stream and unit 10 is assigned to file A. The square brackets [,,4]† indicate the file is resident on disc (at present the only storage medium). 2) If the files are called at execution, they are assigned to the G01. statement (above) thus

```
G01(UNIT5=INPUT,UNIT6=OUTPUT,UNIT10[,,4]=A).
```

and no files are assigned on the PROGRAM card.

d) The use of UPDATE files

The source code of a program can be loaded into an UPDATE file, usually in a POOL, and executed from that file. Modifications to the code can be made at execution, making program development easier. The necessary JCL to create an UPDATE file is:

```
USER(U=123456,AC=A1234)
RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.
PATTACH,SURGE.
UPDATE,N=SURGMOD1,L=124.
GIVE,SURGMOD1,P=SURGE.
7/8/9
  ⌇
program code
  ⌇
```

† the square brackets are non-standard characters on an IBM card punch and must be made up with multipunch characters. [=-ø58,] =-ø58& .

```
    ∫
7/8/9

6/7/8/9
```

i.e. an UPDATE file SURGMOD1 is created and GIVEn to pool SURGE. To execute a program from an UPDATE file, making some corrections to the code, can be done as follows

```
USER(U=123456,AC=A1234)
RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.
PATTACH,SURGE.
UPDATE,P=SURGMOD1,F,L=124.
FORTRAN,I=COMPILE,B=BIN1,O=B.
LOAD,BIN1,CN=GO1.
GO1.
7/8/9
  ∫
program modifications
  ∫
7/8/9
  ∫
data
  ∫
7/8/9
6/7/8/9
```

If there are no program modifications, the 7/8/9 terminator card of the modification section is still required. An UPDATE file cannot be overwritten, so, once modifications are correct, a new UPDATE file, SURGMOD2 say, must be created. This is done by changing the UPDATE card thus,

```
UPDATE,P=SURGMOD1,N=SURGMOD2,F,L=124.
```

and then GIVE the new file SURGMOD2 to the pool. Details of how to modify the code of an UPDATE file are given in the CYBER 200 operating system manual (1980).

e) The use of CONTROLLEE files

A CONTROLLEE file is a loaded object program module which will execute the program on instruction by use of the file name (see subsection b)). Thus, once a program has been satisfactorily debugged, a CONTROLLEE file should be created for future running. Creation of a controllee file is performed at LOAD. If a file is required, say SURGMODC, then the LOAD card becomes

```
LOAD,BIN1,CN=SURGMODC.
```

the controllee is executed by the command

```
SURGMODC.
```

and GIVEn to the pool by

```
GIVE,SURGMODC,P=SURGE.
```

If concurrent I/O is performed in the program, the COMMON blocks must be specified on the LOAD card. These blocks are not stored in the controllee but are called at execution. In fact, it is a useful space saving idea to put any large arrays into COMMON blocks and specify them on the LOAD card. Thus, similar to the LOAD card in subsection b), COMMON blocks are specified

```
LOAD,BIN1,CN=SURGMODC,GROS=*BUFIO,GROS=*BUFIT.
```

where the parameter is now GROS instead of GRSP. All these COMMON blocks are aligned on small page boundaries within the memory.

f Data files

Three types of file are used in the forecasting scheme, formatted (card image), unformatted and undefined structure (for concurrent I/O). Before creating data files, space should be REQUESTed on a disc pack sufficient to cover the size of the file. Assume two files need to be created, file A for storage in a pool and file B to be stored as a permanent file. The REQUEST command is the same for both e.g.

```
       REQUEST,A,RT=x,PACK=DISK01
```

where x = R for formatted file

W for unformatted file

U for undefined structure

and DISK01 is the name of the disc where the data is stored.

Then, once data has been written to the file A it should be given to the pool

```
       GIVE,A,P=SURGE,AC=RW.
```

where RW are read and write permissions. If AC is omitted the default is read only.

To save file B, the statement

```
       DEFINE,B.
```

is used.

If files are not given to a pool or defined they are treated as temporary files and are lost at the end of the job.

g) Loading the surge forecasting suite.

The necessary steps involved are as follows:

i) Run program SETUP to create the model data set, CPSMD3.

ii) Run program METSET to create the file of meteorological interpolation parameters, CSPID1.

iii) Run program SETUPT to create tidal input file, CSPBIT.

iv) Load program GESMOD into controllee file, NCSURGP.

v) Run a small program to initialise, with zero data, the tide and tide plus surge arrays, CPTID1, CPTID2, CPSUR1, CPSUR2.

vi) Create the I/O control file, CPCF1.

vii) Create the model control file, CPCDS.

viii) Create the surge residual port table file, CPTAB1.

ix) Create the file for the residual arrays, CSPRES.

All these files are to be stored on a disc and given to a pool as designated by the Met. Office. In the following JCL, fictitious pool and disc names are specified.

```
i)     USER(U=123456,AC=A1234)
       RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.
       PATTACH,SURGE.
       REQUEST,CPSMD3/36,RT=U,PACK=DISK01.
       FORTRAN,I=COMPILE,B=BIN1,O=B.
       LOAD,BIN1,CN=GO1,GRSP=*BUFIT,GRSP=*BUFRI.
       GO1.
       GIVE,CPSMD3,P=SURGE,AC=RW.
       7/8/9
         ≀
       program SETUP
         ≀
       7/8/9
         ≀
       data for SETUP
         ≀
       7/8/9
```

```
        6/7/8/9
```

On the REQUEST card the file length in blocks is specified immediately after the file name.

```
 ii)    USER(U=123456,AC=A1234)
        RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.
        PATTACH,SURGE.
        PATTACH,POPWAVE.
        REQUEST,CSPID1/56,RT=U,PACK=DISK01.
        FORTRAN,I=COMPILE,B=BIN1,O=B.
        LOAD,BIN1,CN=GO1,GRSP=*HEAD,GRSP=*BUMIT,
        GRSP=*BUMRI.
        GO1.
        GIVE,CSPID1,P=SURGE,AC=RW.
        7/8/9
          ∫
        program METSET
          ∫
        7/8/9
          ∫
        data for METSET
          ∫
        7/8/9
        6/7/8/9
```

Pool POPWAVE is attached to access wind and pressure files WINDFMF and FMFCSP. The LOAD card is continued onto the next line. Any statement can continue on more than one card and is not terminated until a full stop (.) is encountered.

iii)
```
USER(U=123456,AC=A1234)

RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.

PATTACH,SURGE.

REQUEST,CSPBIT/4,RT=W,PACK=DISK01.

FORTRAN,I=COMPILE,B=BIN1,O=B.

LOAD,BIN1,CN=GO1.

GO1.

GIVE,CSPBIT,P=SURGE,AC=RW.

7/8/9
  ∫
program SETUPT
  ∫
7/8/9
  ∫
data for SETUPT
  ∫
7/8/9

6/7/8/9
```

iv)
```
USER(U=123456,AC=A1234)

RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.

PATTACH,SURGE.

FORTRAN,I=COMPILE,B=BIN1,O=B.

LOAD,BIN1,CN=NCSURGP/600,GROS=*BUFIT,

GROS=*BUFRI,GROS=*BUFIO,GROS=*HEADW,GROS=*HEADP,

GROS=*BUFWI,GROS=*BUFPH,GROS=*BUMIT,GROS=*BUMRI,TSP=4.

GIVE,NCSURGP,P=SURGE.

7/8/9
  ∫
```

```
        }
      program GESMOD
        }
      7/8/9

      6/7/8/9
```

No request card is included. The number of blocks required to execute the controllee file must be specified after the file name if greater than the system default value.

```
v)    USER(U=123456,AC=A1234)

      RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.

      PATTACH,SURGE.

      REQUEST,CPTID1/32,RT=U,PACK=DISK01.

      REQUEST,CPTID2/32,RT=U,PACK=DISK01.

      REQUEST,CPSUR1/32,RT=U,PACK=DISK01.

      REQUEST,CPSUR2/32,RT=U,PACK=DISK01.

      FORTRAN,I=COMPILE,B=BIN1,O=B.

      LOAD,BIN1,CN=GO1,GRSP=*BUFIO.

      GO1.

      GIVE,CPTID1,P=SURGE,AC=RW.

      GIVE,CPTID2,P=SURGE,AC=RW.

      GIVE,CPSUR1,P=SURGE,AC=RW.

      GIVE,CPSUR2,P=SURGE,AC=RW.

      7/8/9
        }
      program INIT
        }
      7/8/9

      6/7/8/9
```

vi)
```
USER(U=1234356,AC=A1234)
RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.
PATTACH,SURGE.
REQUEST,CPCF1/4,RT=W,PACK=DISK01.
FORTRAN,I=COMPILE,B=BIN1,O=B.
LOAD,BIN1,CN=GO1.
GO1.
GIVE,CPCF1,P=SURGE,AC=RW.
7/8/9
   ∫
program CONTROL
   ∫
7/8/9
6/7/8/9
```

vii)
```
USER(U=123456,AC=A1234)
RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.
PATTACH,SURGE.
REQUEST,CPCDS/4,RT=R,PACK=DISK01.
COPY,INPUT,CPCDS.
COPY,CPCDS,OUTPUT.
GIVE,CPCDS,P=SURGE,AC=RW.
7/8/9
   ∫
data for GESMOD
   ∫
7/8/9
6/7/8/9
```

```
viii) USER(U=123456,AC=A1234)
 & ix) RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.
       PATTACH,SURGE.
       REQUEST,CPTAB1/12,RT=W,PACK=DISK01.
       REQUEST,CSPRES/324,RT=U,PACK=DISK01.
       GIVE,CPTAB1,P=SURGE,AC=RW.
       GIVE,CSPRES,P=SURGE,AC=RW.
       7/8/9
       6/7/8/9
```

Here, null files with a set space allocation are given to the pool.

h) Running the surge forecasting suite

With all datasets stored in the pool and the program controllee file created, running the suite is a simple matter. The following JCL will invoke the controllee file and produce a surge forecast defined by the parameters set up in CPCDS.

```
USER(U=123456,AC=A1234)
RESOURCE,TL=20,WS=1000,LP=5,JCAT=CS.
PATTACH,SURGE.
NCSURGP.
7/8/9
6/7/8/9
```

When the suite is run in operational mode at the Meteorological Office, all files in the pool are copied across into the operational pool and executed from there. This is done by Meteorological Office personnel.

8. Concluding remarks.

The programs and procedures outlined in this report have been in operation at the Meteorological Office since September 1982. An indication of the speed attainable with the new computer can be obtained by comparing the CPU time required for forecasts using the old and new schemes. The old scheme, using CSM, required ~ 90 seconds on the IBM 360/195. The new version, CSX, which has $\sim 40\%$ more calculation points and requires a shorter timestep and hence 25% more iterations, takes ~ 12 seconds on the CDC CYBER 205E. The optimisation work on the new code produced a reduction in CPU time required by a factor ~ 4, compared with a standard FORTRAN version, not allowing for synchronous input/output. The programs and model(s) may be modified in between surge seasons or even within a season if such modifications are likely to improve the quality of the forecasts.

9. Acknowledgements.

We would like to express our thanks to Jim Ephraums and Vic Blackman at the Meteorological Office without whose assistance these programs would probably not have been written! This work was funded by MAFF.

10. References.

CARTWRIGHT, D.E., EDDEN, A.C., SPENCER, R. & VASSIE, J.M. 1980 The tides of the northeast Atlantic Ocean.
Philosophical Transactions of the Royal Society of London, A, 298, 87-139.

C.D.C. CYBER 200 Fortran version 2, 1981. Reference Manual.

C.D.C. CYBER 200 operating System, version 1, 1980. Reference Manual.

DICKINSON, A. 1982 Optimizing numerical weather forecasting models for the CRAY-1 and CYBER 205 computers. Computer Physics
Computer Physics Communications, 26, 459-468.

DOODSON, A.T. 1921 The harmonic development of the tide-generating potential.
Proceedings of the Royal Society of London, A, 100, 305-329.

DOODSON, A.T. & WARBURG, H.D. 1941 Admiralty manual of tides.
London: HMSO. 270pp.

FLATHER, R.A. 1979 Recent results from a storm surge prediction scheme for the North Sea.
pp 385-409 in, Marine Forecasting, (ed. J.C.J. Nihoul) Amsterdam: Elsevier. 493pp.

FLATHER, R.A. 1980 Results from a model of the north east Atlantic relating to the Norwegian Coastal current.
pp 427-458 in vol.2, The Norwegian Coastal Current (ed. R. Saetre & M. Mork). Bergen University. 795pp.

FLATHER, R.A. 1981 Practical surge prediction using numerical models.
pp 21-43 in, Floods due to winds and high tides, (ed. D.H. Peregrine). London: Academic Press. 109pp.

FLATHER, R.A. & HEAPS, N.S. 1975 Tidal computations for Morecambe Bay.
Geophysical Journal of the Royal Astronomical Society, 42, 489-517.

FLATHER, R.A. & DAVIES, A.M. 1978 On the specification of meteorological forcing in numerical models for North Sea storm surge prediction, with application to the surge of 2 to 4 January 1976.
Deutsche Hydrographische Zeitschrift, Erganzungsheft, A, No.15, 51pp.

FLATHER, R.A. & PROCTOR, R. 1983 Prediction of North Sea storm surges using numerical models: Recent developments in the U.K.
pp 299-317 in, North Sea Dynamics, (ed. J. Sundermann & W. Lenz), Berlin: Springer, Verlag. 693pp.

SMITH, S.D. & BANKE, E.G. 1975 Variation of the sea surface drag coefficient with wind speed.
Quarterly Journal of the Royal Meteorological Society, 101, 665-673.

TOWNSEND, J. 1981 Storm surges and their forecasting.
pp 1-7 in, Floods due to winds and high tides, (ed. D.H. Peregrine). London: Academic Press. 109pp.

2 vector pipelines

Data streamed from memory
through pipes and back to
memory for each vector instruction

Triadic statement of one scalar
and two vectors executed as one
instruction

1 million 64-bit words of memory

Memory is bit addressable

64-bit or 32-bit arithmetic

Separate scalar and vector
processors

256 64-bit registers

20 nanosecond cycle time

6 on-line discs, each disc having
140,000 x 512 words capacity

Front-ended by CYBER 18

Maximum vector length 65535

Table 2. Some features of the CYBER 205E

STORM SURGE FORECAST C.S.X.   DATA STARTS AT 12 HRS GMT 27/1/1983   RESIDUAL ELEVATIONS IN M

TIME	POINT NUMBER														
GMT	585	548	726	991	1214	1258	1349	1436	1479	1484	1441	1267	1227	1186	966
	STWY	WICK	ABDN	NSHL	IMMI	WASH	LOFT	WALT	SEND	VLIS	HOEK	DHEL	BORK	CUXN	ESBG
1200	.33	.56	.42	.24	.28	.18	-.01	-.23	-.25	-.01	.19	.44	1.12	1.26	1.29
1300	.28	.60	.54	.37	.34	.34	.07	-.14	-.28	-.05	.11H	.42L	.81	1.26	1.12H
1400	.27	.65	.60	.40H	.33	.37	.15	-.03	-.23	-.01	-.01	.31	.47	.98	1.01
1500	.29	.51	.65	.42	.37	.37	.26L	.06	-.12	.03	.04	.27	.33L	.66	1.09
1600	.21	.43L	.67	.55	.39	.38	.29	.11	.01	.13	.15	.32	.41	.54	1.09
1700	.11	.52	.59	.70	.39H	.35H	.32	.18L	.11	.24	.26	.38	.68	.68L	1.04
1800	.19H	.52	.56L	.75	.52	.36	.35	.28	.18L	.29L	.34	.53H	.83	.86	.95
1900	.26	.50	.52	.78	.69	.53	.50	.36	.29	.36	.44L	.56	.80	.89	.84L
2000	.26	.41	.56	.65L	.91	.71	.59H	.42	.38	.41	.41	.55	.77H	.88	.64
2100	.15	.39	.49	.57	1.01	.85	.68	.50	.44	.49	.46	.60	.77	.87	.63
2200	.13	.30H	.45	.65	.95	.92	.81	.58	.50	.58	.59	.69	.79	.89	.74
2300	.21	.26	.43	.77	.87L	.91	.85	.66H	.57H	.62	.73	.77	.76	.83H	.86
0	.19L	.38	.43H	.69	.80	.82L	.80	.77	.66	.70	.78	.85	.77	.83	.91
100	.15	.30	.38	.47	.79	.77	.85	.90	.78	.80H	.79	.85L	.86	.87	.92H
200	.17	.38	.37	.39H	.68	.82	.87	.83	.83	.76	.77H	.83	.92	.93	.92
300	.16	.36	.28	.27	.57	.83	.88L	.79	.80	.74	.79	.81	.99	.97	.89
400	.16	.25L	.18	.25	.54	.66	.85	.75	.72	.74	.81	.89	1.03L	1.06	.82
500	.16	.17	.17	.23	.38H	.41H	.68	.71L	.65	.64	.76	.88	.99	1.11L	.86
600	.19H	.21	.17L	.17	.22	.20	.59	.76	.65L	.60	.66	.81H	.93	1.08	1.01
700	.17	.25	.24	.14	.09	.10	.61	.69	.75	.70L	.60L	.72	.96	1.19	1.15L
800	.22	.32	.23	.10	.02	.04	.50H	.58	.80	.67	.59	.69	.98	1.28	1.19
900	.25	.37	.25	.12L	.05	-.02	.38	.56	.72	.60	.61	.62	.92H	1.20	1.09
1000	.26	.34H	.26	.17	.08	.01	.27	.46	.54	.51	.56	.64	.92	1.11	1.04
1100	.19	.30	.30	.17	.16L	.07	.22	.25H	.30	.40	.48	.58	.90	1.07H	1.16
1200	.16	.26	.25H	.18	.20	.14L	.30	.10	.02H	.32	.37	.50	.73	.97	1.29
1300	.22L	.22	.32	.24	.18	.20	.27	.12	-.12	.24H	.25	.50L	.57	.86	1.24H
1400	.28	.23	.26	.29	.13	.19	.25	.13	-.08	.24	.26H	.48	.58	.84	1.10
1500	.33	.24	.19	.24H	.10	.14	.20L	.14	.05	.23	.29	.44	.69	.79	.93
1600	.36	.26	.13	.18	.15	.08	.15	.15	.13	.16	.22	.39	.72L	.74	.82
1700	.38	.24L	.13	.09	.16H	.08	.13	.14L	.13	.16	.17	.35	.62	.71	.73
1800	.41H	.26	.03	-.01	.14	.16H	.18	.14	.09	.20	.16	.25	.45	.69L	.69
1900	.40	.20	.01L	-.07	.04	.12	.17	.06	.03L	.22L	.19L	.17H	.33	.66	.72L
2000	.48	.06	.10	-.04	-.11	-.06	.11	.06	-.03	.24	.27	.200	.28	.57	.72
2100	.59	.18	.12	-.02L	-.18	-.19	.03H	.06	-.06	.28	.28	.29	.31H	.46	.66
2200	.48	.27	.09	-.05	-.15	-.24	.01	.01	-.03	.22	.23	.33	.36	.36	.59
2300	.46	.36H	.10	.07	-.09	-.24	.06	.00	-.02	.22	.25	.41	.41	.36H	.52
2400	.49	.46	.20	.08	.01	-.20	.08	.01	-.05	.25	.27	.45	.59	.45	.57

Table 1. STWS forecast table

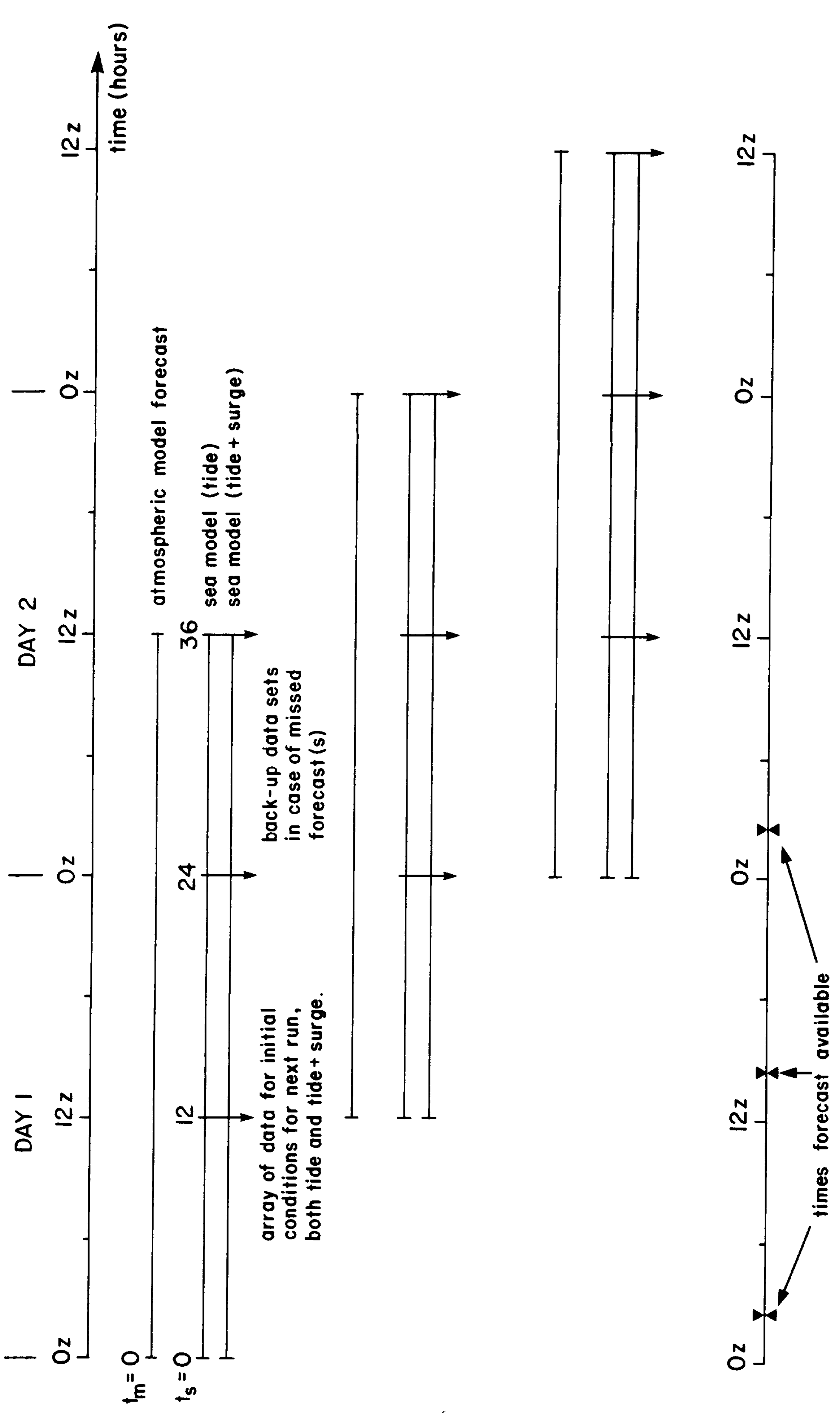

DAY 1
DAY 2
0z
12z
0z
12z
0z
12z
time (hours)
$t_m = 0$
$t_s = 0$
12
24
36
atmospheric model forecast
sea model (tide)
sea model (tide + surge)
array of data for initial conditions for next run, both tide and tide + surge.
back-up data sets in case of missed forecast(s)
times forecast available
0z
12z
0z
12z
0z
12z

Figure 1. Schematic of the storm surge forecasting procedure

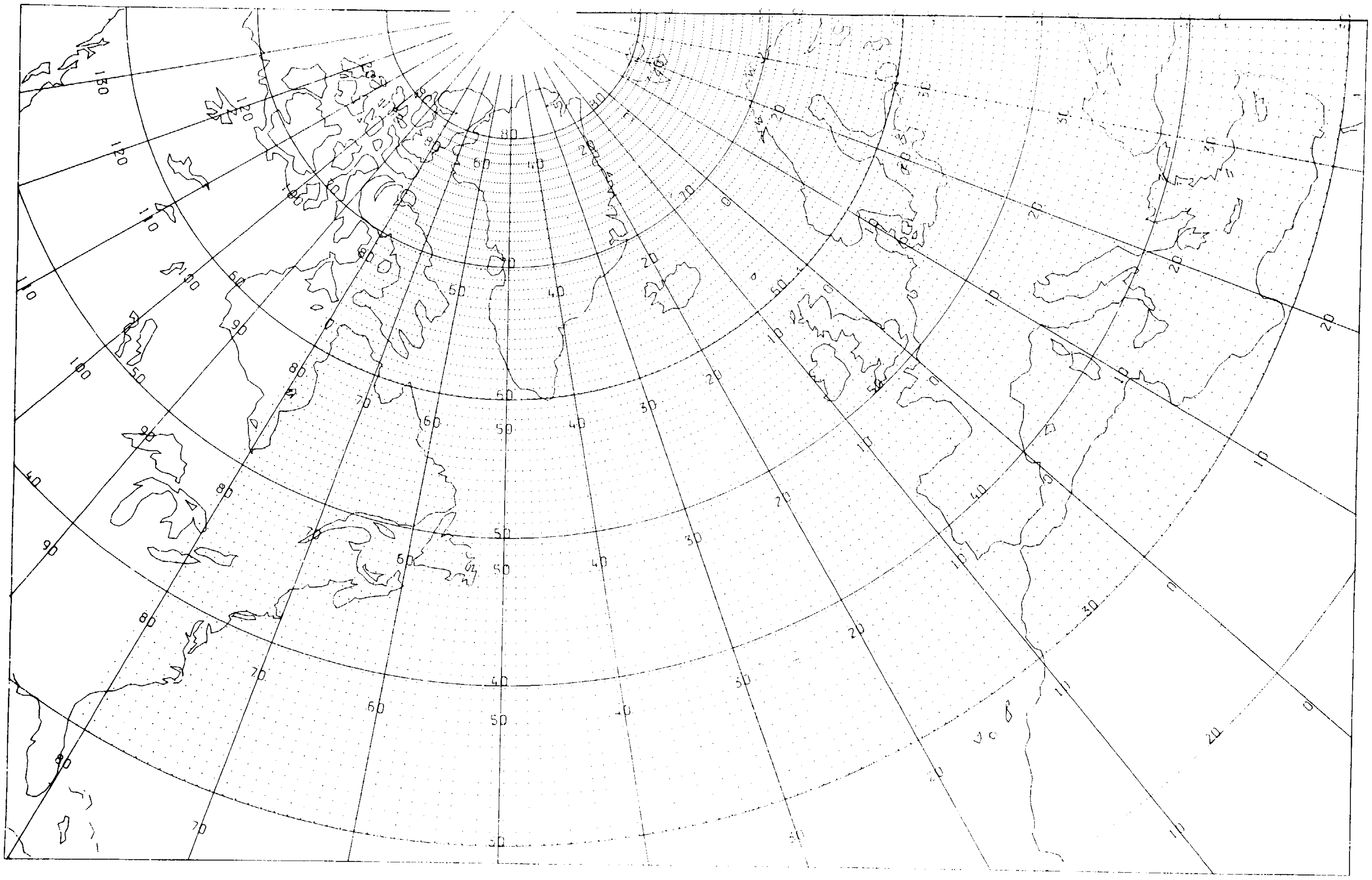


Figure 2. Grid of the Met Office 15 level atmospheric model showing air pressure points

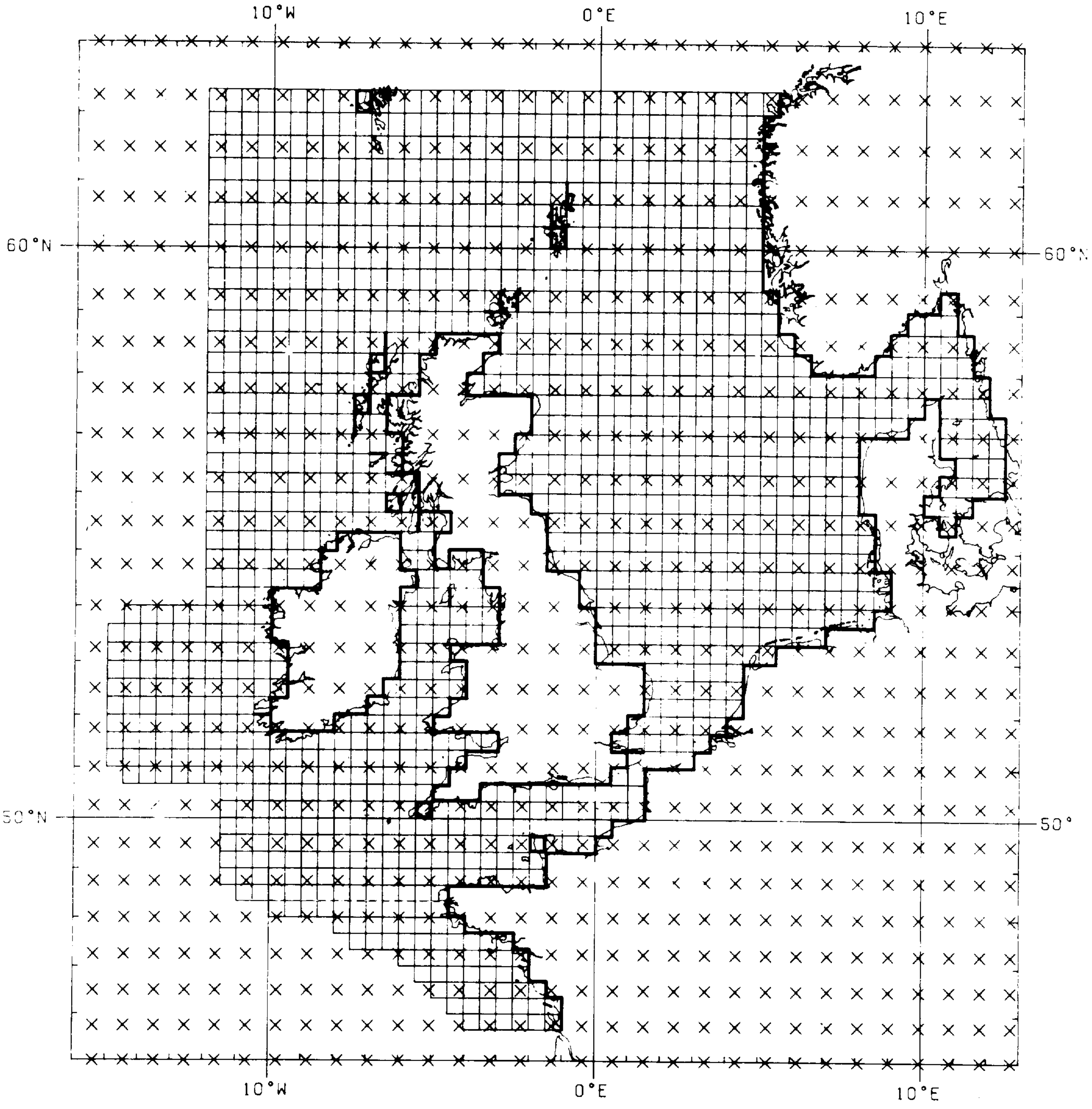


Figure 3. Grid of sea model CSX showing pressure points (X) of the Met Office 15 level atmospheric model

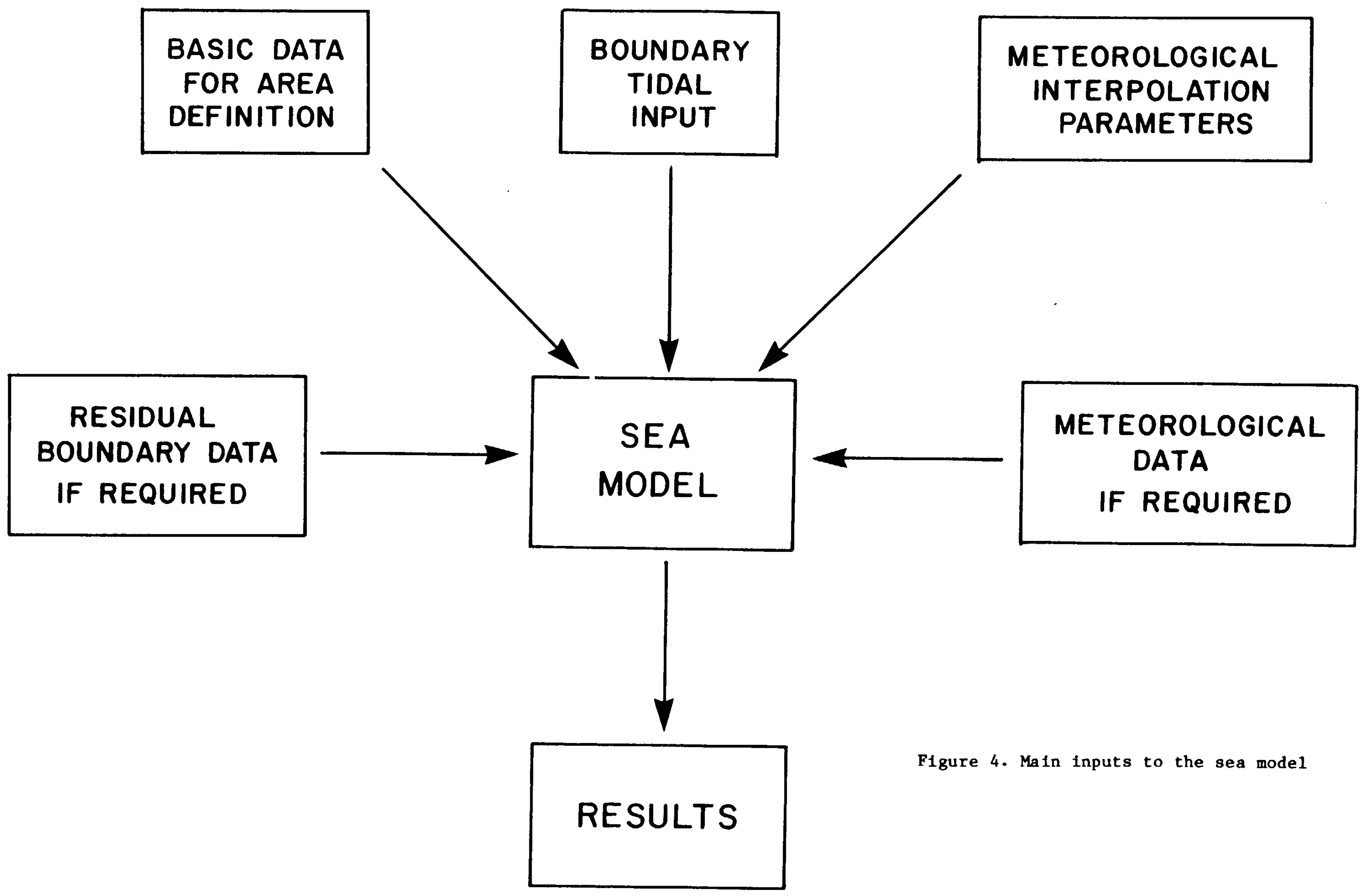

BASIC DATA FOR AREA DEFINITION
BOUNDARY TIDAL INPUT
METEOROLOGICAL INTERPOLATION PARAMETERS
RESIDUAL BOUNDARY DATA IF REQUIRED
SEA MODEL
METEOROLOGICAL DATA IF REQUIRED
RESULTS

Figure 4. Main inputs to the sea model

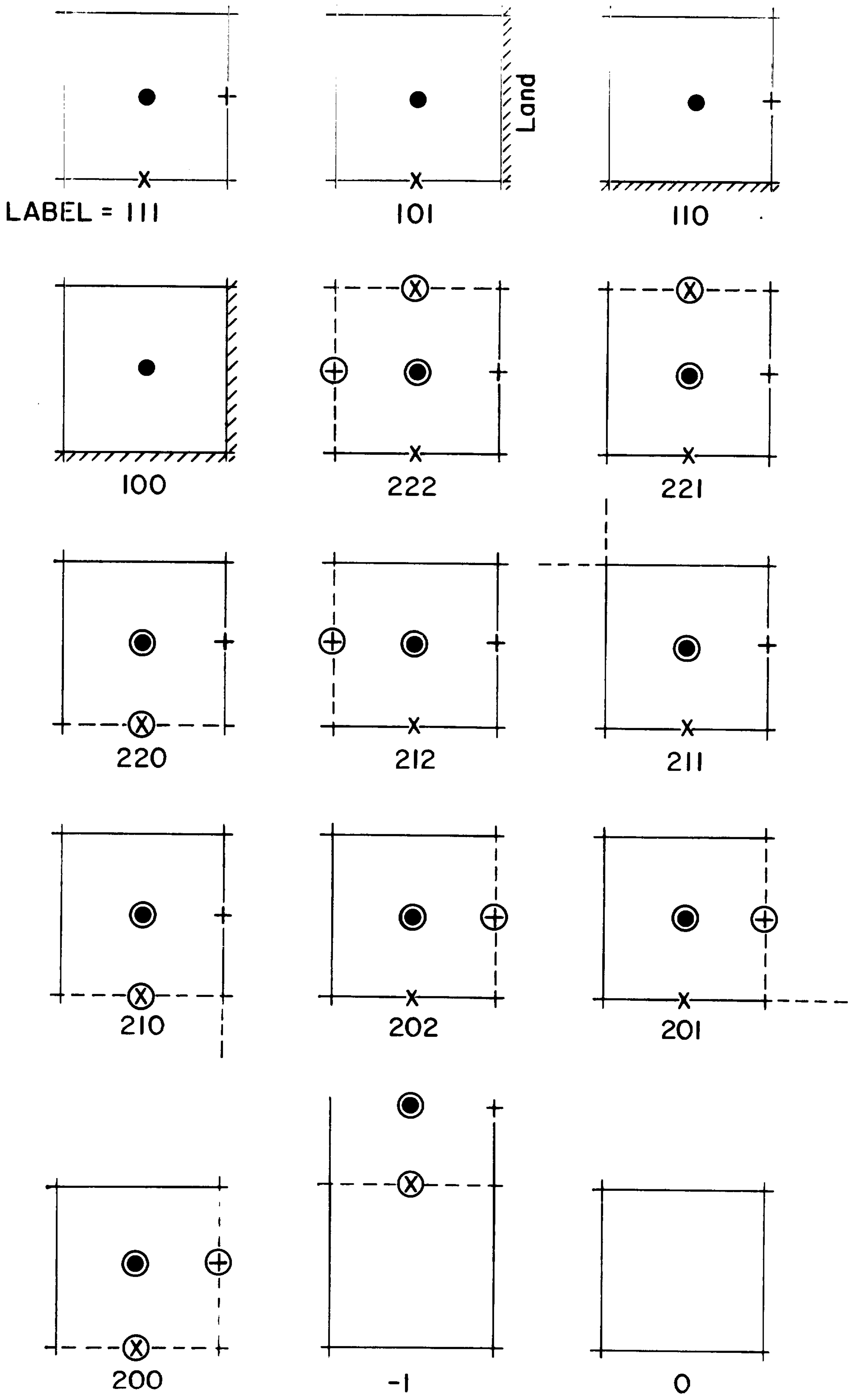


Figure 5. Labels for element identification

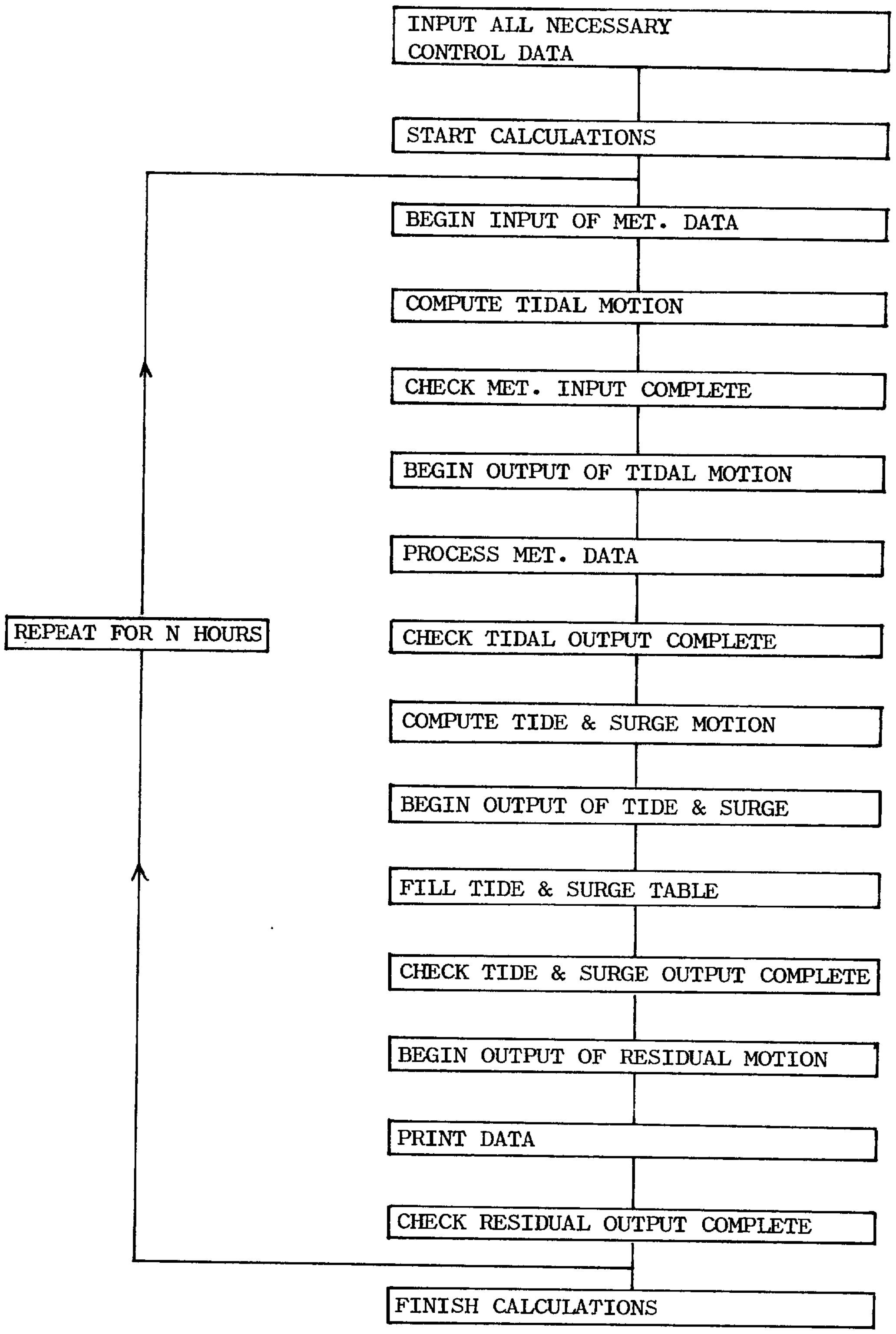

INPUT ALL NECESSARY CONTROL DATA
START CALCULATIONS
BEGIN INPUT OF MET. DATA
COMPUTE TIDAL MOTION
CHECK MET. INPUT COMPLETE
BEGIN OUTPUT OF TIDAL MOTION
PROCESS MET. DATA
REPEAT FOR N HOURS
CHECK TIDAL OUTPUT COMPLETE
COMPUTE TIDE & SURGE MOTION
BEGIN OUTPUT OF TIDE & SURGE
FILL TIDE & SURGE TABLE
CHECK TIDE & SURGE OUTPUT COMPLETE
BEGIN OUTPUT OF RESIDUAL MOTION
PRINT DATA
CHECK RESIDUAL OUTPUT COMPLETE
FINISH CALCULATIONS

Figure 6 - General layout of sea model computations with emphasis on I/O of data

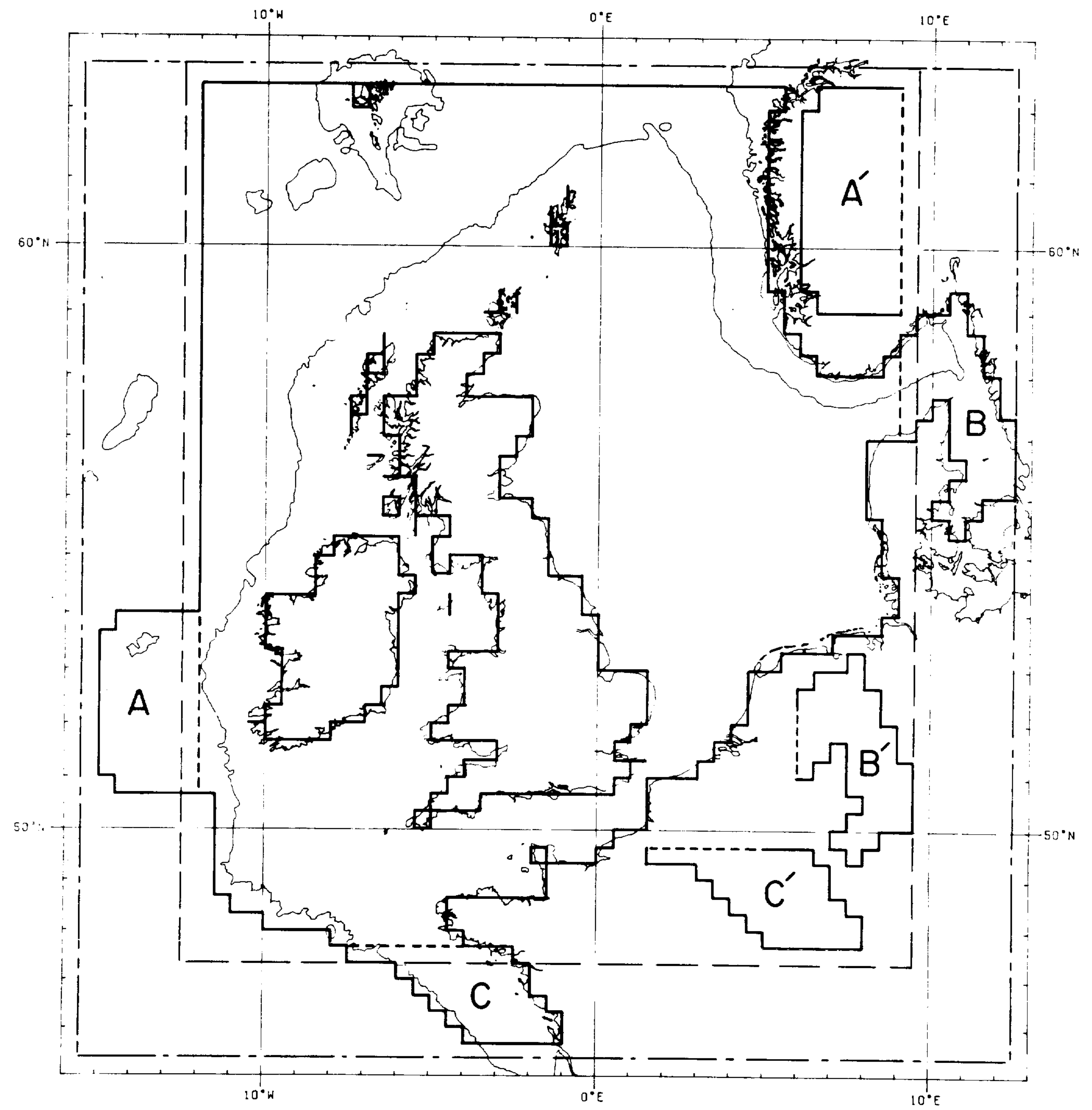
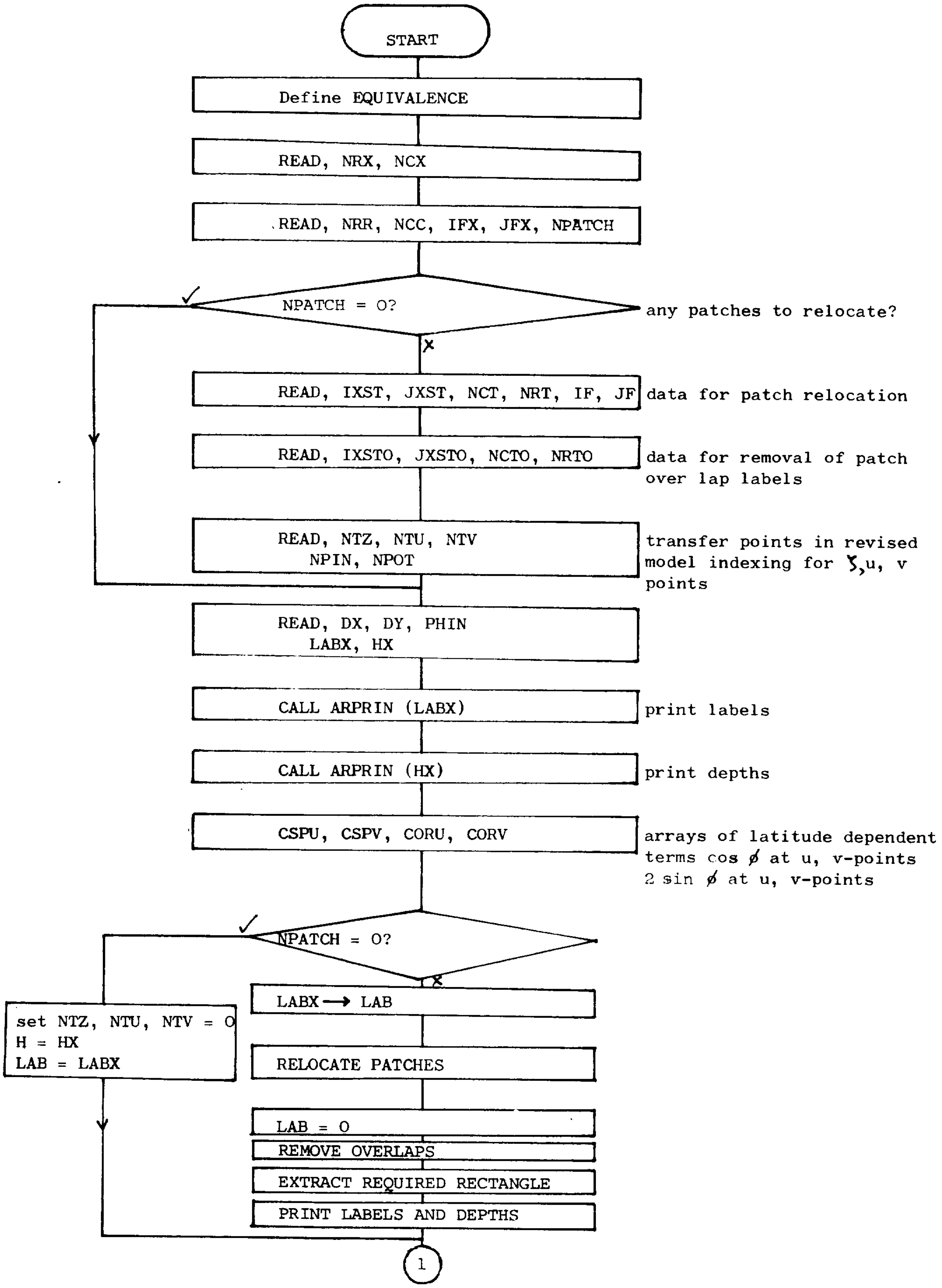


Figure 7. Sea model outline showing patch relocation

Figure 8   Flow chart for program SETUP


START
Define EQUIVALENCE
READ, NRX, NCX
READ, NRR, NCC, IFX, JFX, NPATCH
NPATCH = O?
✓
any patches to relocate?
X
READ, IXST, JXST, NCT, NRT, IF, JF
data for patch relocation
READ, IXSTO, JXSTO, NCTO, NRTO
data for removal of patch over lap labels
READ, NTZ, NTU, NTV NPIN, NPOT
transfer points in revised model indexing for ζ,u, v points
READ, DX, DY, PHIN LABX, HX
CALL ARPRIN (LABX)
print labels
CALL ARPRIN (HX)
print depths
CSPU, CSPV, CORU, CORV
arrays of latitude dependent terms cos ϕ at u, v-points
2 sin ϕ at u, v-points
NPATCH = O?
✓
X
set NTZ, NTU, NTV = O
H = HX
LAB = LABX
LABX ⟶ LAB
RELOCATE PATCHES
LAB = O
REMOVE OVERLAPS
EXTRACT REQUIRED RECTANGLE
PRINT LABELS AND DEPTHS
1

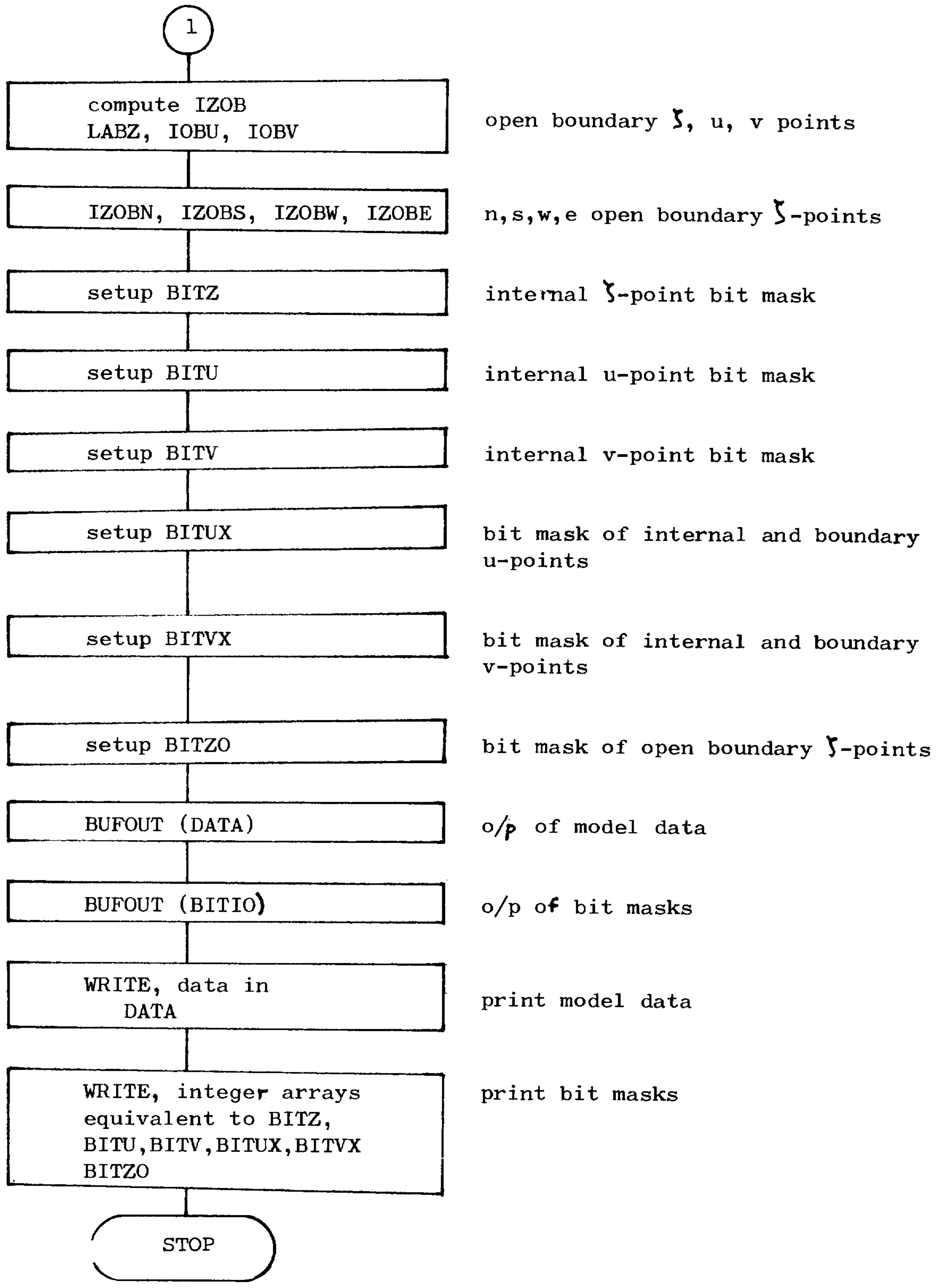

1
compute IZOB LABZ, IOBU, IOBV
open boundary ζ, u, v points
IZOBN, IZOBS, IZOBW, IZOBE
n,s,w,e open boundary ζ-points
setup BITZ
internal ζ-point bit mask
setup BITU
internal u-point bit mask
setup BITV
internal v-point bit mask
setup BITUX
bit mask of internal and boundary u-points
setup BITVX
bit mask of internal and boundary v-points
setup BITZO
bit mask of open boundary ζ-points
BUFOUT (DATA)
o/p of model data
BUFOUT (BITIO)
o/p of bit masks
WRITE, data in DATA
print model data
WRITE, integer arrays equivalent to BITZ, BITU,BITV,BITUX,BITVX BITZO
print bit masks
STOP

Figure 8 continued

```
00000000000000000000000000000000000000000000000000000000
00000000111111111011111111111111111111111100000000000000
00000000111111111111111111111111111111111000000000000000
00000000111111111111111111111111111111111000000000000000
00000000111111111111111111111111111111111000000000000000
00000000111111111111111111111111111111111000000000000000
00000000111111111111111111111111111111111000000000000000
00000000111111111111111111110111111111111000000000000000
00000000111111111111111111111111111111111000000000000000
00000000111111111111111111111111111111111000000000000000
00000000111111111111111111111111111111111000000000001000
00000000111111111111111111111111111111111100000000111000
00000000111111111111110000111111111111111100000001111100
00000000111111111101100001111111111111111100001111111100
00000000111111111111100011111111111111111111111111111110
00000000111111111010000000011111111111111111111111101110
00000000111111111100000000011111111111111111111111001111
00000000111111111111000000011111111111111111111000001111
00000000111111111111000000111111111111111111111000000111
00000000111111111111100000111111111111111111111000001111
00000000111111111101000000011111111111111111111000011100
00000000111111111111111100000111111111111111111100001000
00000000111111110000110000000111111111111111111100000000
00000000111111100000110110000111111111111111111100000000
00000000111111100000011110000001111111111111111110000000
00000000111000000000111111000001111111111111111110000000
00011111111100000000111111000000111111111111111110000000
00111111111100000000111111000000111111111111100000000000
00111111111110000000111000000000111111111100000000000000
00111111111110000000111000000000000111110000000000000000
00111111111110000001111000000000000111110000000000000000
00111111111100000011111000000000000111110000000000000000
00111111111100001111110000000000001111110000000000000000
00111111111111111111111111000000011111100000000000000000
00111111111111111111111100000000001111000000000000000000
00111111111111111111111000000000011000000000000000000000
00000000011111111111110001111111111000000000000000000000
00000000011111111111101111111111111000000000000000000000
00000000011111111111111111111111100000000000000000000000
00000000011111111111111111110111000000000000000000000000
00000000011111111111111111111000000000000000000000000000
00000000011111111111111111111000000000000000000000000000
00000000001111111111110000000000000000000000000000000000
00000000000011111111110000000000000000000000000000000000
00000000000000001111111100000000000000000000000000000000
00000000000000000111111111000000000000000000000000000000
00000000000000000000111111100000000000000000000000000000
00000000000000000000011111110000000000000000000000000000
00000000000000000000001111111000000000000000000000000000
00000000000000000000000111111100000000000000000000000000
00000000000000000000000011111100000000000000000000000000
00000000000000000000000000000000000000000000000000000000
```

Figure 9 : Bit mask for internal ζ -points

```
00000000000000000000000000000000000000000000
01111111110111111111111111111111111100111110
01111111111111111111111111111111111001111110
01111111111111111111111111111111111001111110
01111111111111111111111111111111111001111110
01111111111111111111111111111111111001111110
01111111111111111111111111111111111001111110
01111111111111111111110111111111111001111110
01111111111111111111111111111111111001111110
01111111111111111111111111111111111001111110
01111111111111111111111111111111111100111110
01111111111111111111111111111111111100000000
01111111111111100001111111111111111110000000
01111111111011000011111111111111111111000010
01111111111111000111111111111111111111111110
01111111110100000000011111111111111111111110
01111111111100000000011111111111111111111110
01111111111110000000111111111111111111111000
01111111111110000001111111111111111111111000
01111111111111000001111111111111111111111000
01111111111101000000011111111111111111111000
01111111111111110000001111111111111111111100
01111111100001100000001111111111111111111100
01111111000001101100001111111111111111111100
01111111000000111100000011111111111111111110
01111000000001111100000011111111111111111110
01111000000001111100000001111111111111111100
01111000000001111100000001111111111111100000
01111100000001110000000001111111111100001000
01111100000001110000000000001111110000111000
01111100000011111000000000001111110001111100
01111000000111110000000000001111110001111100
01111000011111100000000000011111100001111110
01111111111111111110000000111111000001101110
01111111111111111100000000011110000001001111
01111111111111111000000000110000000000001111
00111111111111100011111111110000000000000111
00111111111111011111111111110000000000001111
00111111111111111111111111000000000000011100
00111111111111111111101110001111111111001000
00111111111111111111110000000001111111100000
00111111111111111111110000000000111111100000
00011111111111110000000000000000011111110000
00000111111111110000000000000000001111111000
00000000011111111000000000000000000111111000
00000000000000000000000000000000000000000000
```

Figure 10 : Bit mask for internal ζ-points after patch relocation

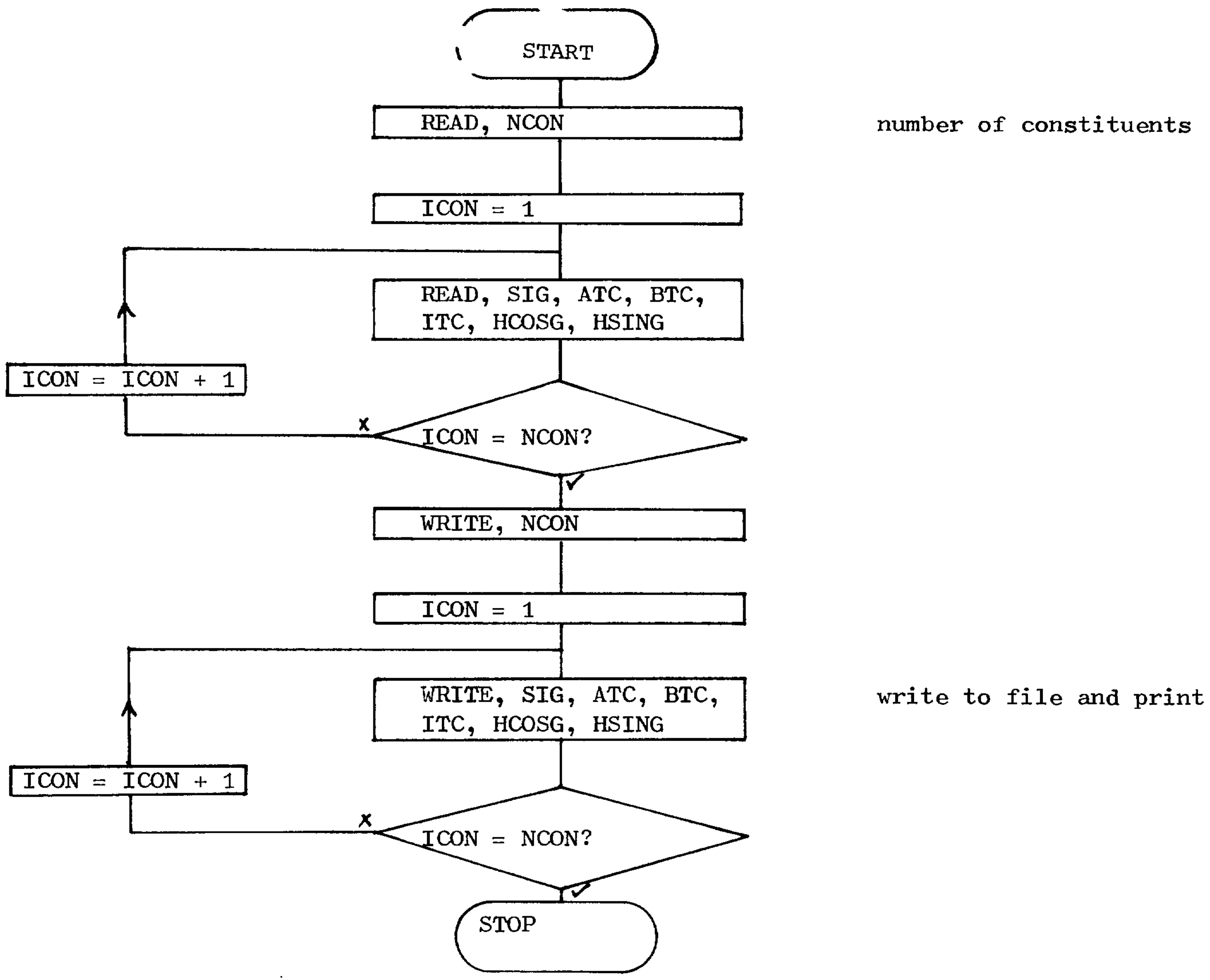


Figure 11 - Flow chart for program SETUPT

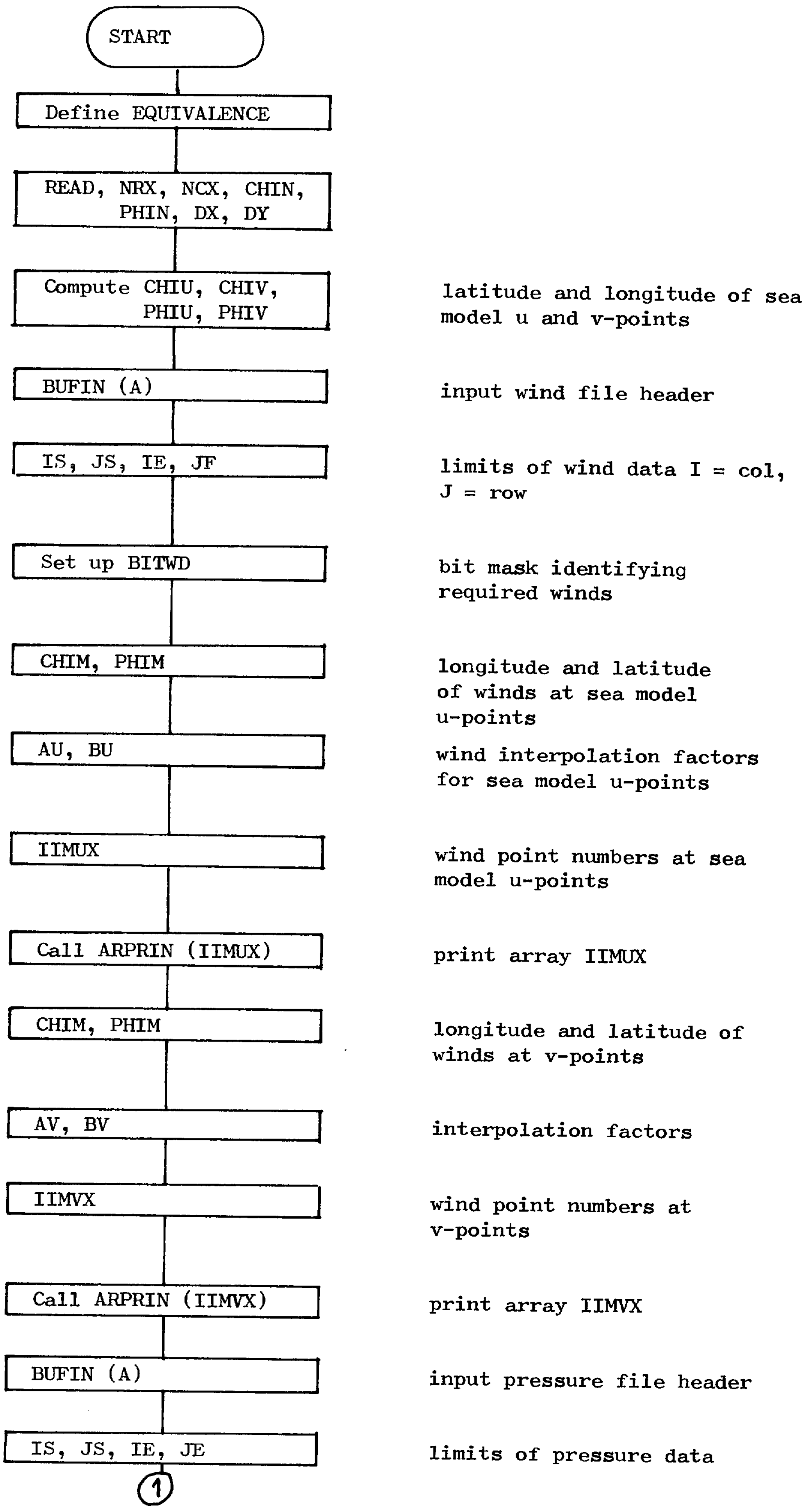

START
Define EQUIVALENCE
READ, NRX, NCX, CHIN, PHIN, DX, DY
Compute CHIU, CHIV, PHIU, PHIV
latitude and longitude of sea model u and v-points
BUFIN (A)
input wind file header
IS, JS, IE, JF
limits of wind data I = col, J = row
Set up BITWD
bit mask identifying required winds
CHIM, PHIM
longitude and latitude of winds at sea model u-points
AU, BU
wind interpolation factors for sea model u-points
IIMUX
wind point numbers at sea model u-points
Call ARPRIN (IIMUX)
print array IIMUX
CHIM, PHIM
longitude and latitude of winds at v-points
AV, BV
interpolation factors
IIMVX
wind point numbers at v-points
Call ARPRIN (IIMVX)
print array IIMVX
BUFIN (A)
input pressure file header
IS, JS, IE, JE
limits of pressure data
1

Figure 12 - Flow chart for program METSET

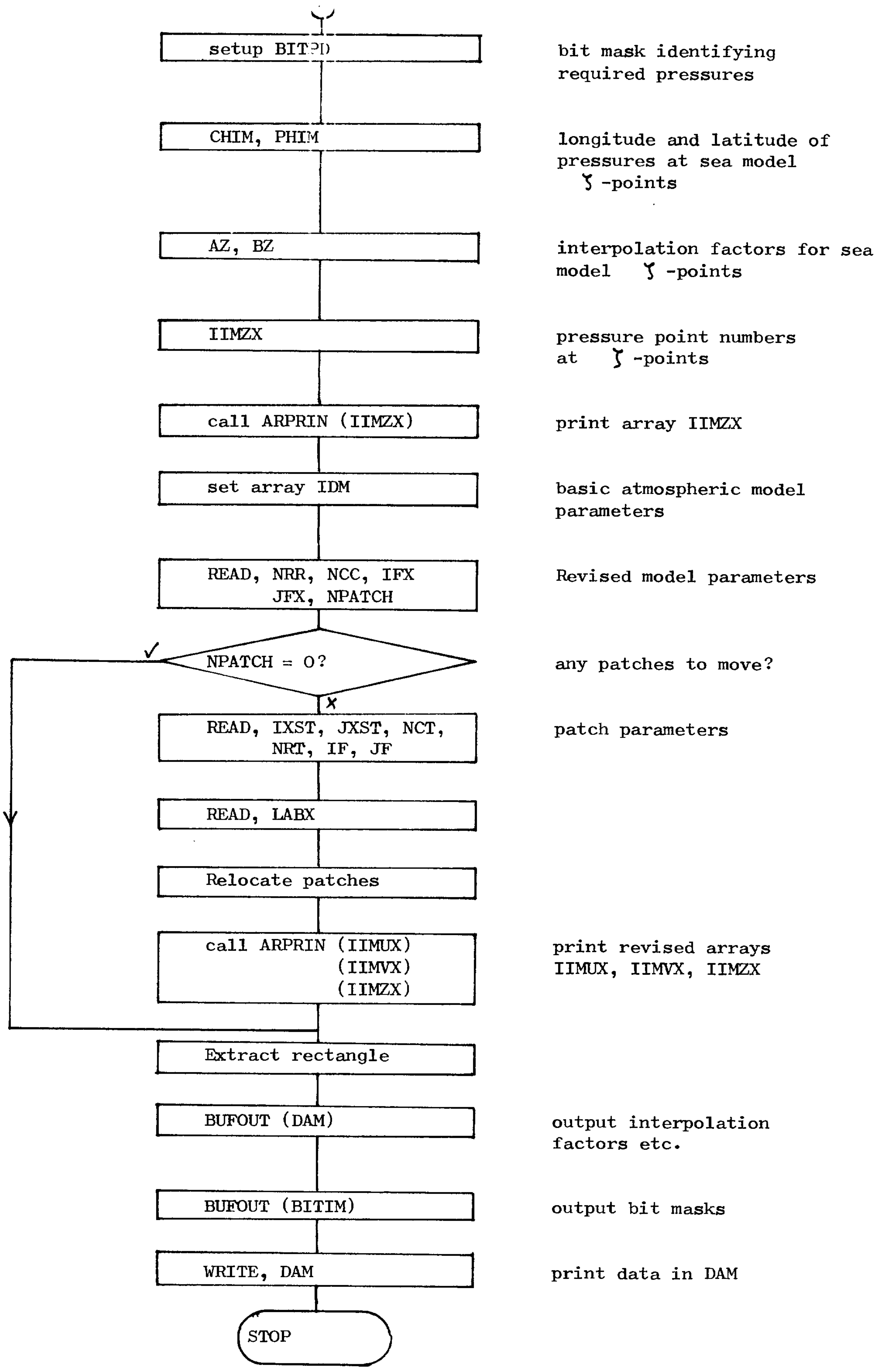

setup BITPD
bit mask identifying required pressures
CHIM, PHIM
longitude and latitude of pressures at sea model ζ -points
AZ, BZ
interpolation factors for sea model ζ -points
IIMZX
pressure point numbers at ζ -points
call ARPRIN (IIMZX)
print array IIMZX
set array IDM
basic atmospheric model parameters
READ, NRR, NCC, IFX JFX, NPATCH
Revised model parameters
NPATCH = 0?
any patches to move?
✓
✗
READ, IXST, JXST, NCT, NRT, IF, JF
patch parameters
READ, LABX
Relocate patches
call ARPRIN (IIMUX) (IIMVX) (IIMZX)
print revised arrays IIMUX, IIMVX, IIMZX
Extract rectangle
BUFOUT (DAM)
output interpolation factors etc.
BUFOUT (BITIM)
output bit masks
WRITE, DAM
print data in DAM
STOP

Figure 12 continued

Subroutine ARPRIN - print an integer or real array

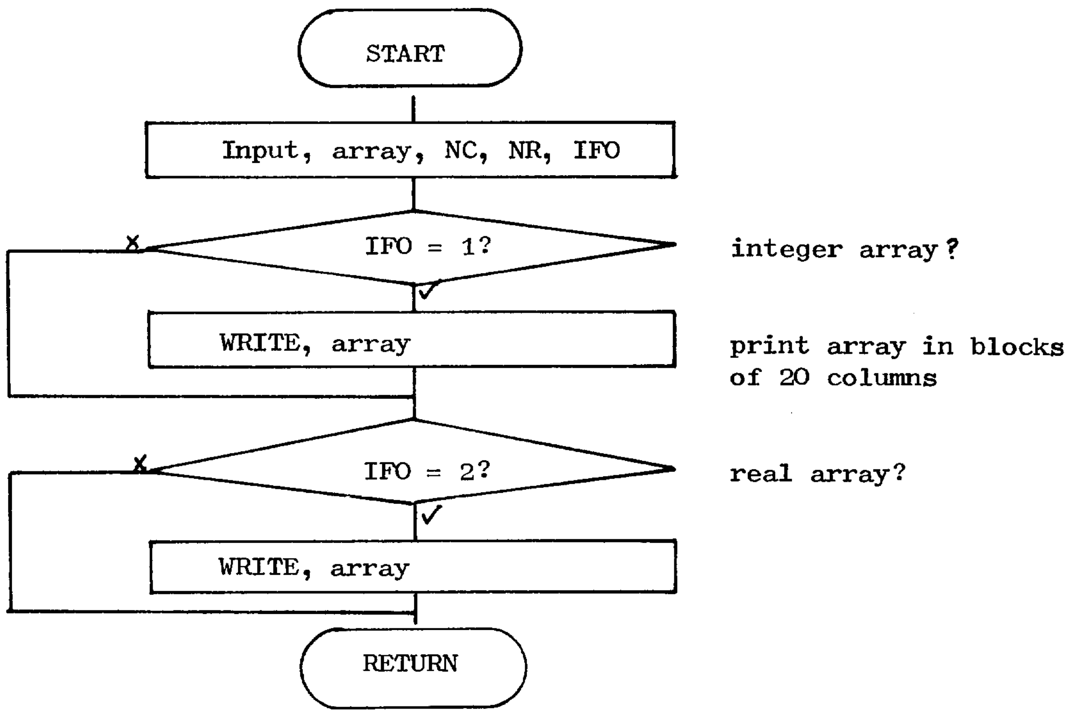


Figure 12 continued

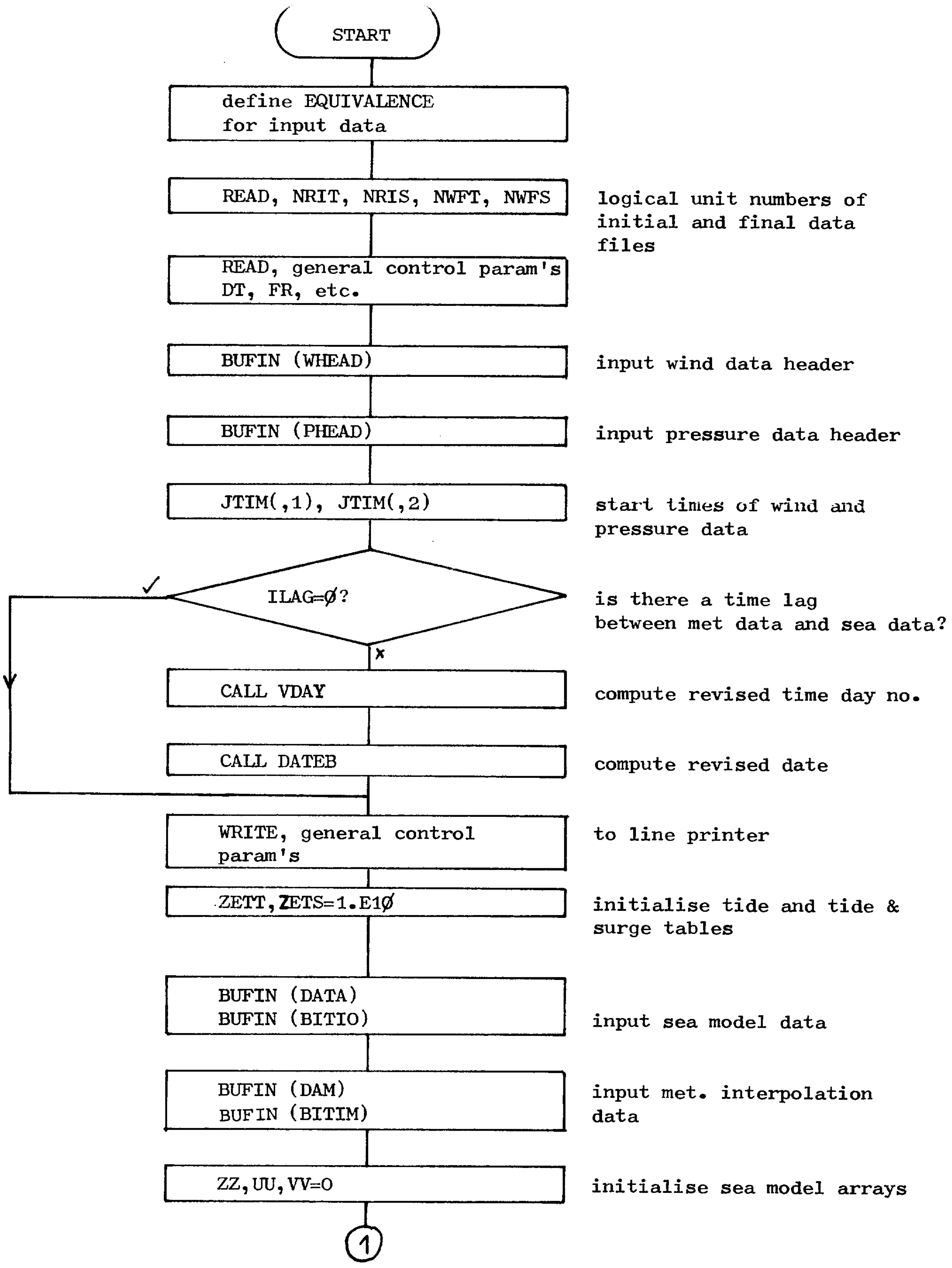

START
define EQUIVALENCE for input data
READ, NRIT, NRIS, NWFT, NWFS
logical unit numbers of initial and final data files
READ, general control param's DT, FR, etc.
BUFIN (WHEAD)
input wind data header
BUFIN (PHEAD)
input pressure data header
JTIM(,1), JTIM(,2)
start times of wind and pressure data
ILAG=Ø?
✓
x
is there a time lag between met data and sea data?
CALL VDAY
compute revised time day no.
CALL DATEB
compute revised date
WRITE, general control param's
to line printer
ZETT,ZETS=1.E1Ø
initialise tide and tide & surge tables
BUFIN (DATA)
BUFIN (BITIO)
input sea model data
BUFIN (DAM)
BUFIN (BITIM)
input met. interpolation data
ZZ,UU,VV=0
initialise sea model arrays
1

Figure 13 - Flowchart for program GESMOD

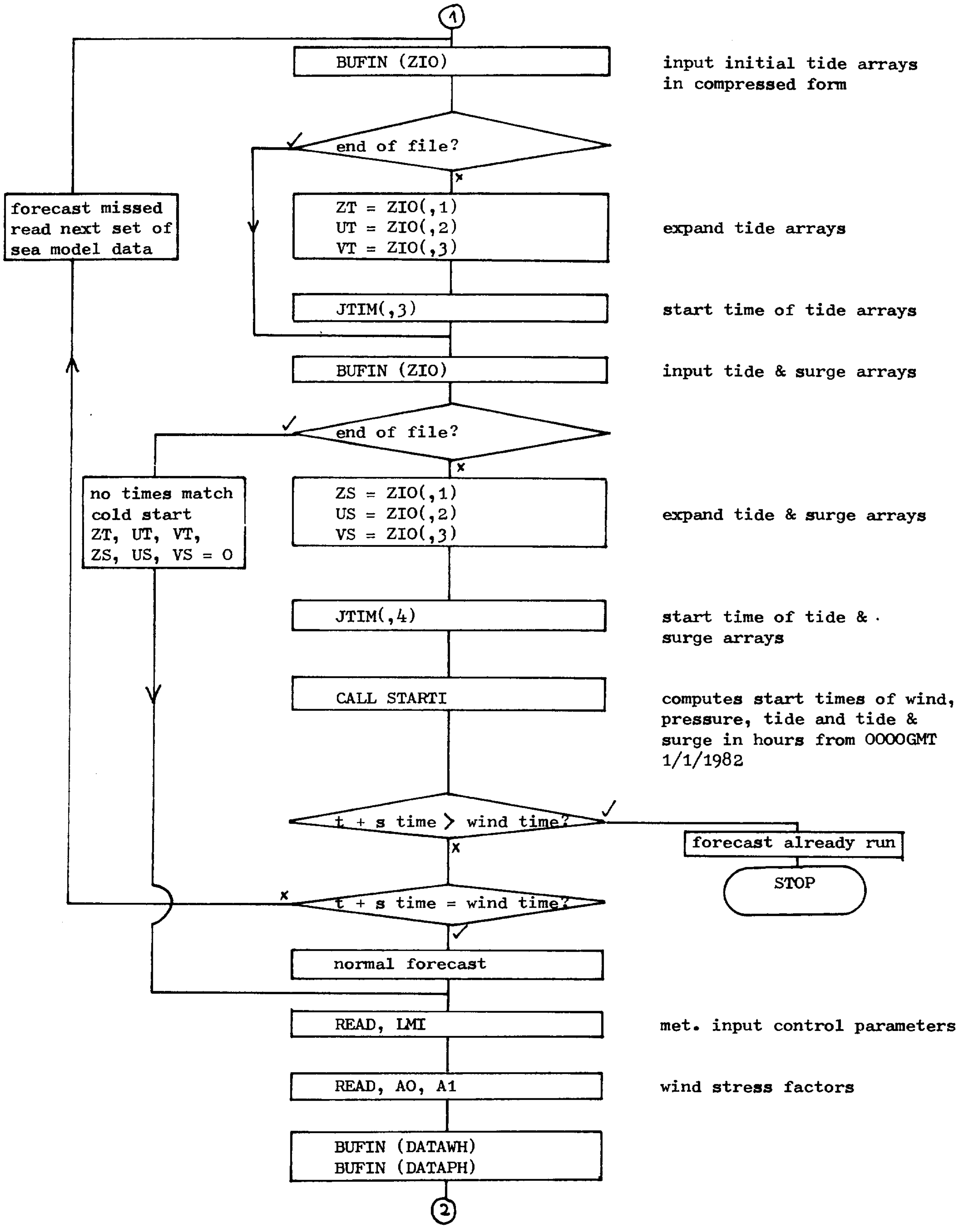

1
BUFIN (ZIO)
input initial tide arrays in compressed form
end of file?
✓
x
ZT = ZIO(,1)
UT = ZIO(,2)
VT = ZIO(,3)
expand tide arrays
forecast missed read next set of sea model data
JTIM(,3)
start time of tide arrays
BUFIN (ZIO)
input tide & surge arrays
end of file?
✓
x
no times match cold start ZT, UT, VT, ZS, US, VS = 0
ZS = ZIO(,1)
US = ZIO(,2)
VS = ZIO(,3)
expand tide & surge arrays
JTIM(,4)
start time of tide & surge arrays
CALL STARTI
computes start times of wind, pressure, tide and tide & surge in hours from 0000GMT 1/1/1982
t + s time > wind time?
✓
forecast already run
STOP
x
t + s time = wind time?
x
✓
normal forecast
READ, LMI
met. input control parameters
READ, AO, A1
wind stress factors
BUFIN (DATAWH)
BUFIN (DATAPH)
2

Figure 13 continued

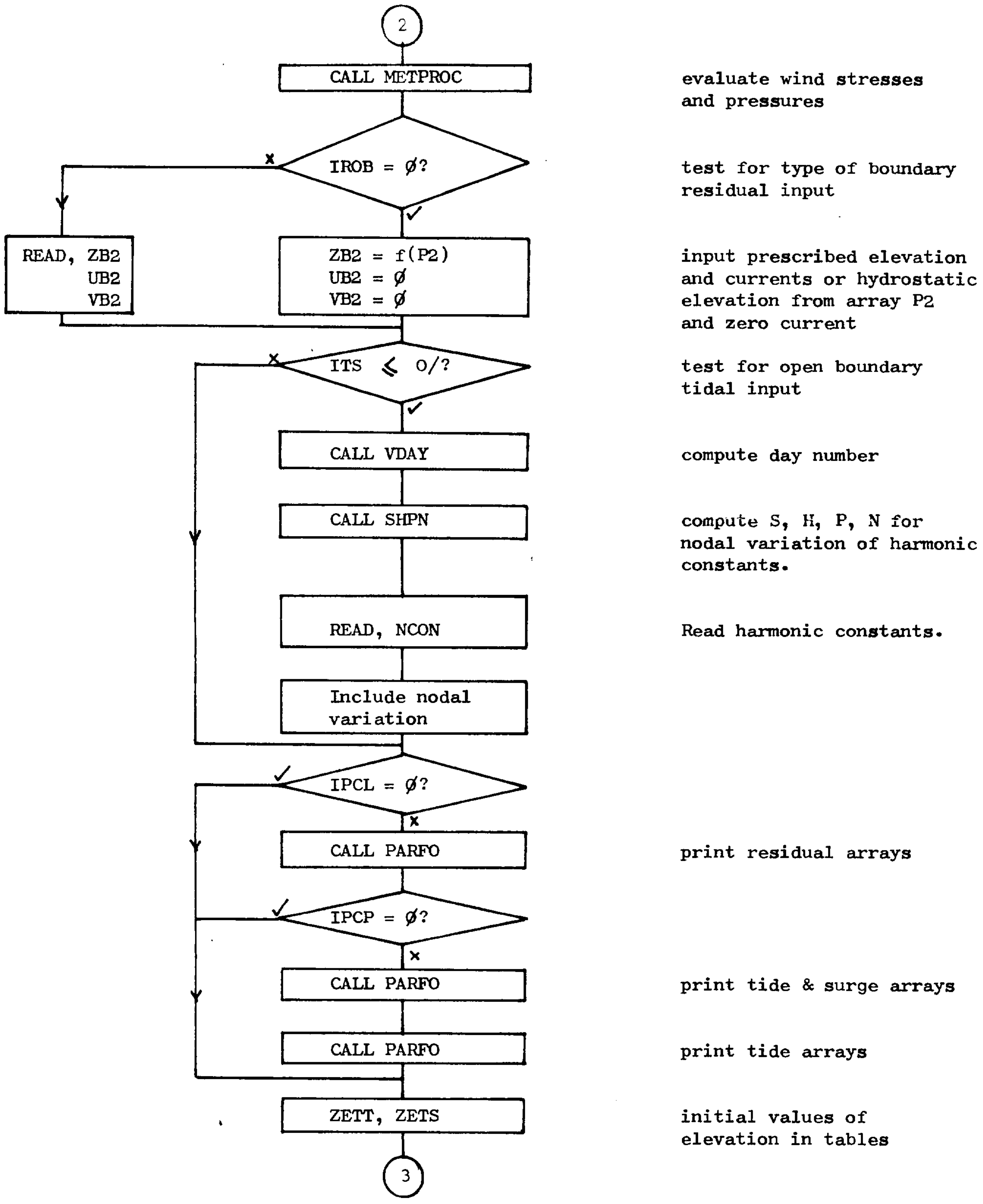

2
CALL METPROC
evaluate wind stresses and pressures
IROB = Ø?
x
✓
test for type of boundary residual input
READ, ZB2 UB2 VB2
ZB2 = f(P2) UB2 = Ø VB2 = Ø
input prescribed elevation and currents or hydrostatic elevation from array P2 and zero current
ITS ⩽ 0/?
x
✓
test for open boundary tidal input
CALL VDAY
compute day number
CALL SHPN
compute S, H, P, N for nodal variation of harmonic constants.
READ, NCON
Read harmonic constants.
Include nodal variation
IPCL = Ø?
✓
x
CALL PARFO
print residual arrays
IPCP = Ø?
✓
x
CALL PARFO
print tide & surge arrays
CALL PARFO
print tide arrays
ZETT, ZETS
initial values of elevation in tables
3

Figure 13 continued

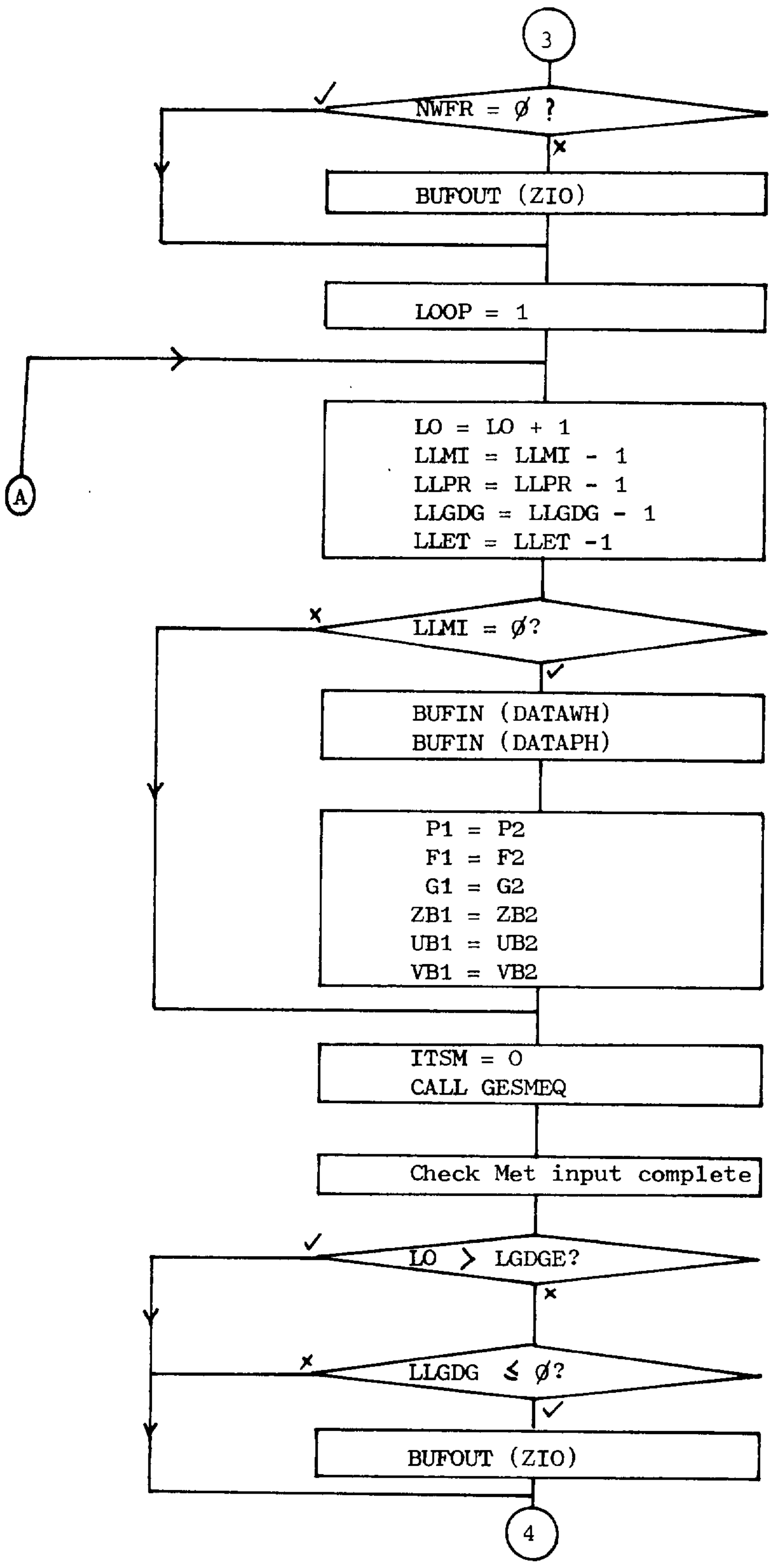
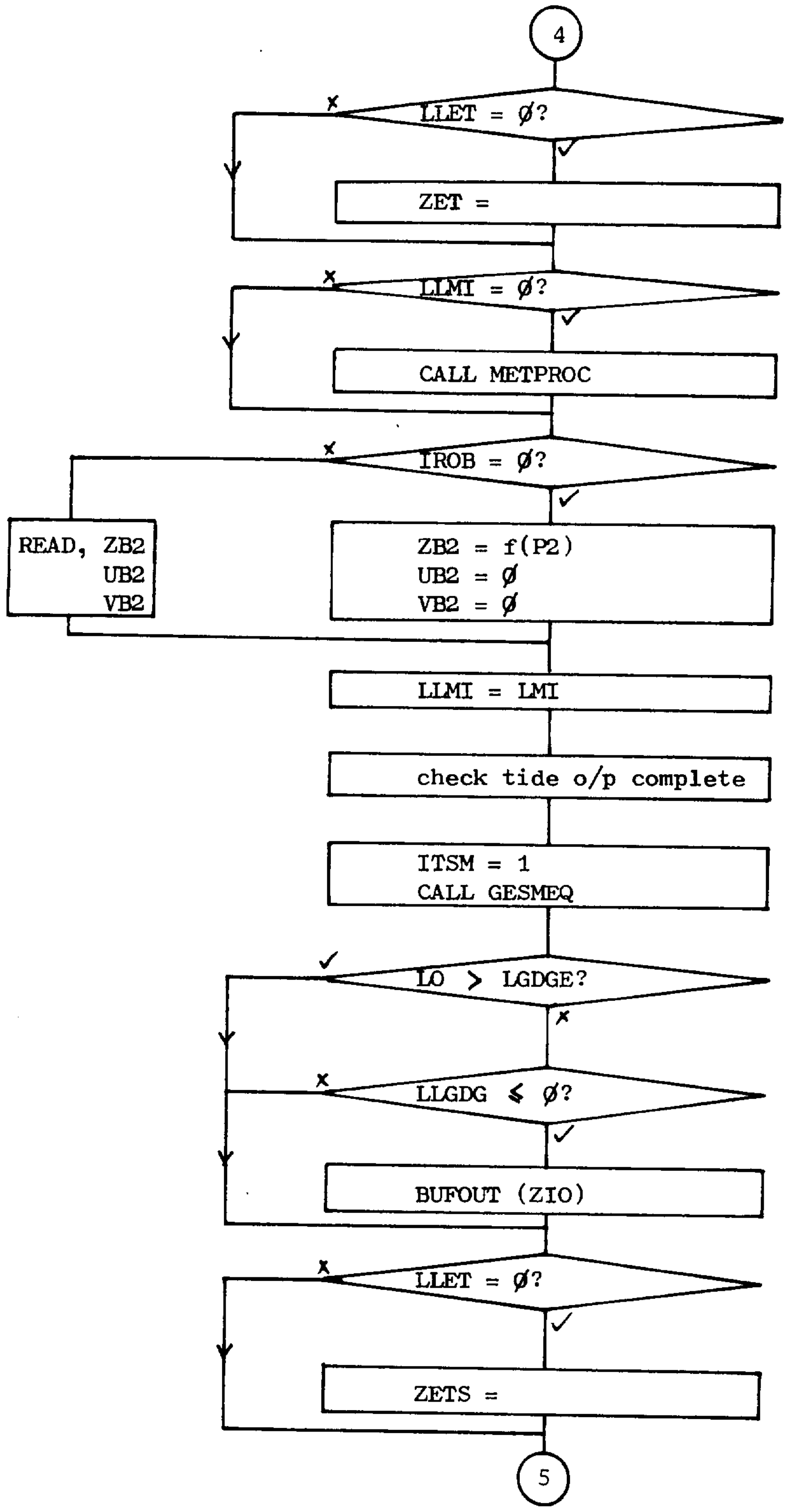


Figure 13 continued



fill tide tables?

perform met processing?

obtain boundary residual input

reset met counter

integrate for tide & surge

output tide & surge arrays

fill tide & surge tables?

Figure 13 continued

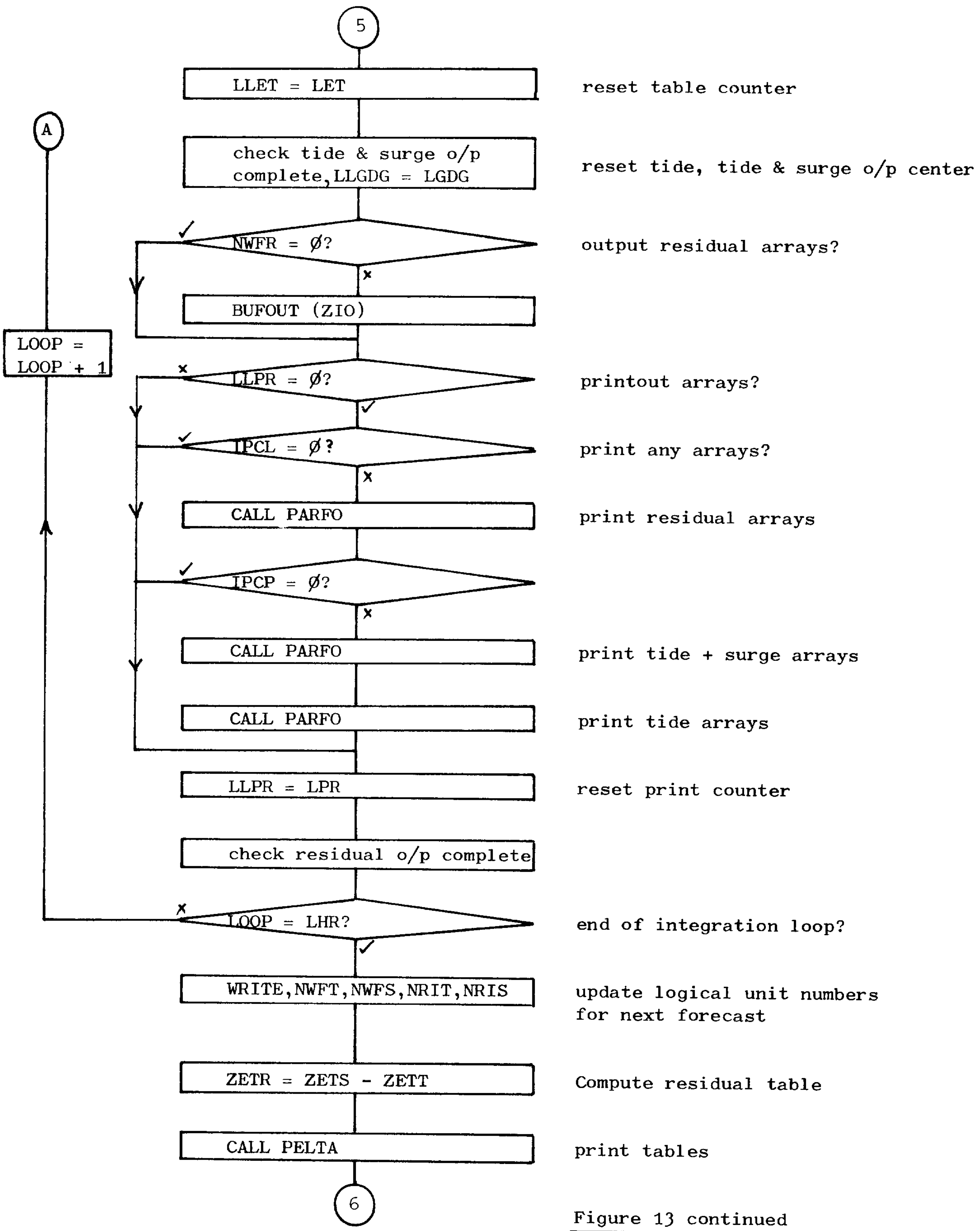
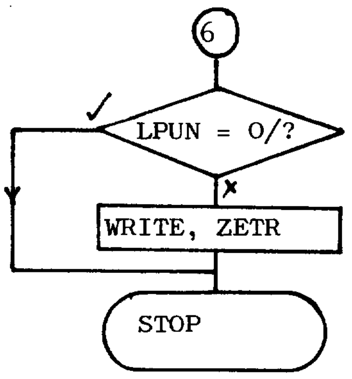


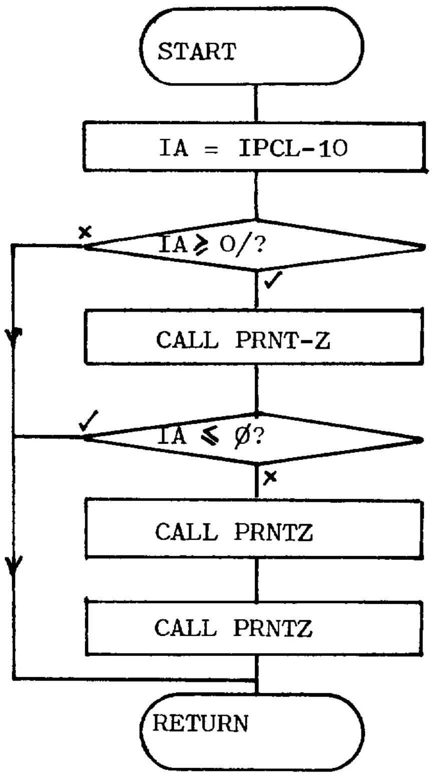
Figure 13 continued



o/p of tables?

Figure 13 continued

Subroutine PARFO - prepare arrays for printing



Subroutine PRNTZ - print array

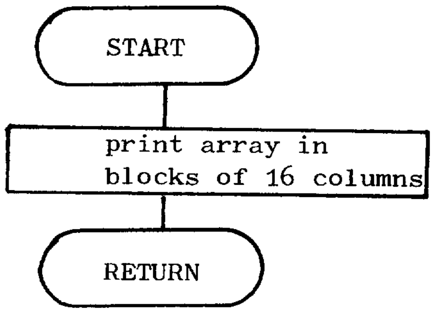


Figure 13 continued

subroutine PELTA - print elevation table

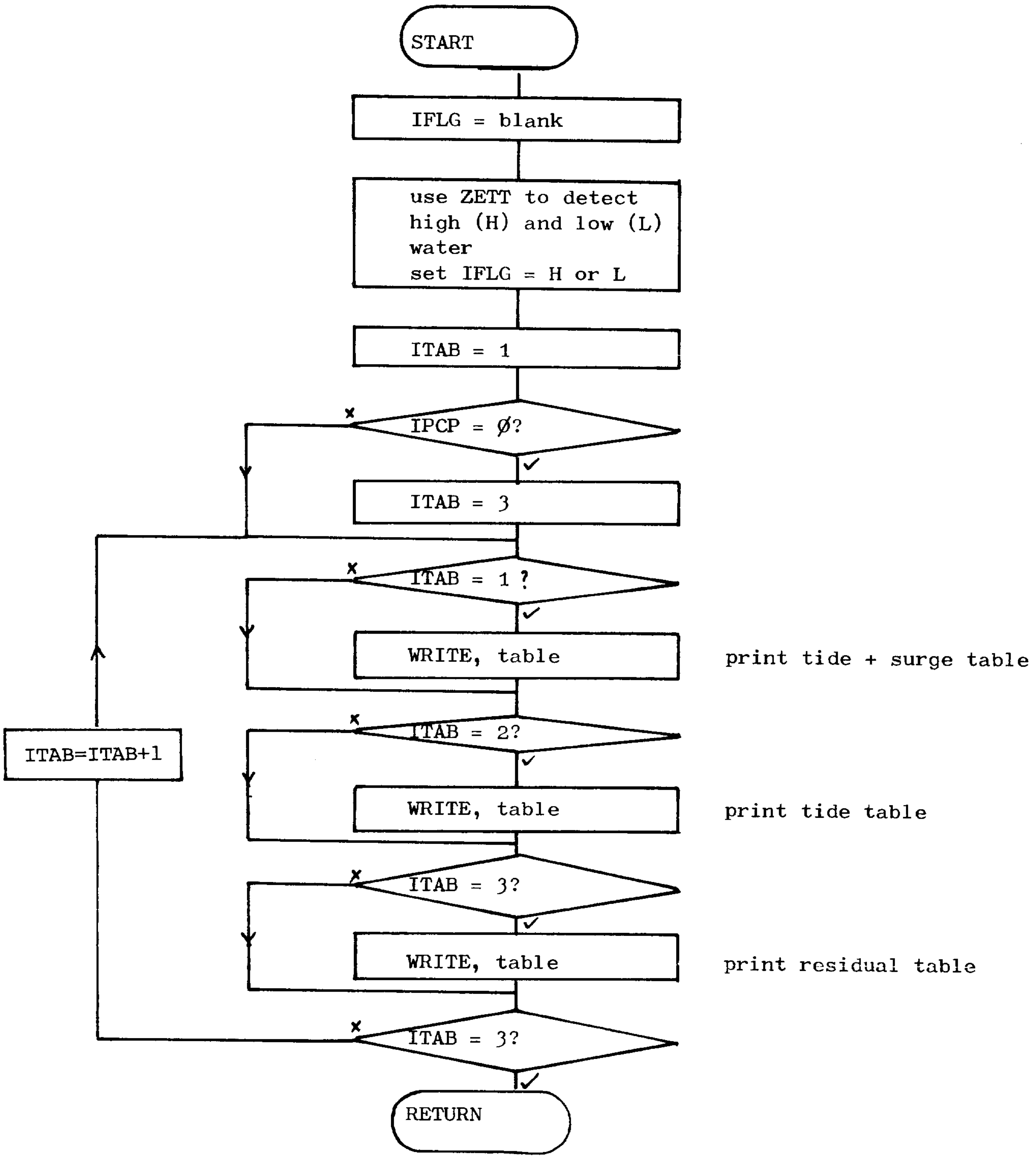


Figure 13 continued

Subroutine STARTI - compute elapsed times in hours for meteorological and sea model data since 0000 GMT 1/1/1982

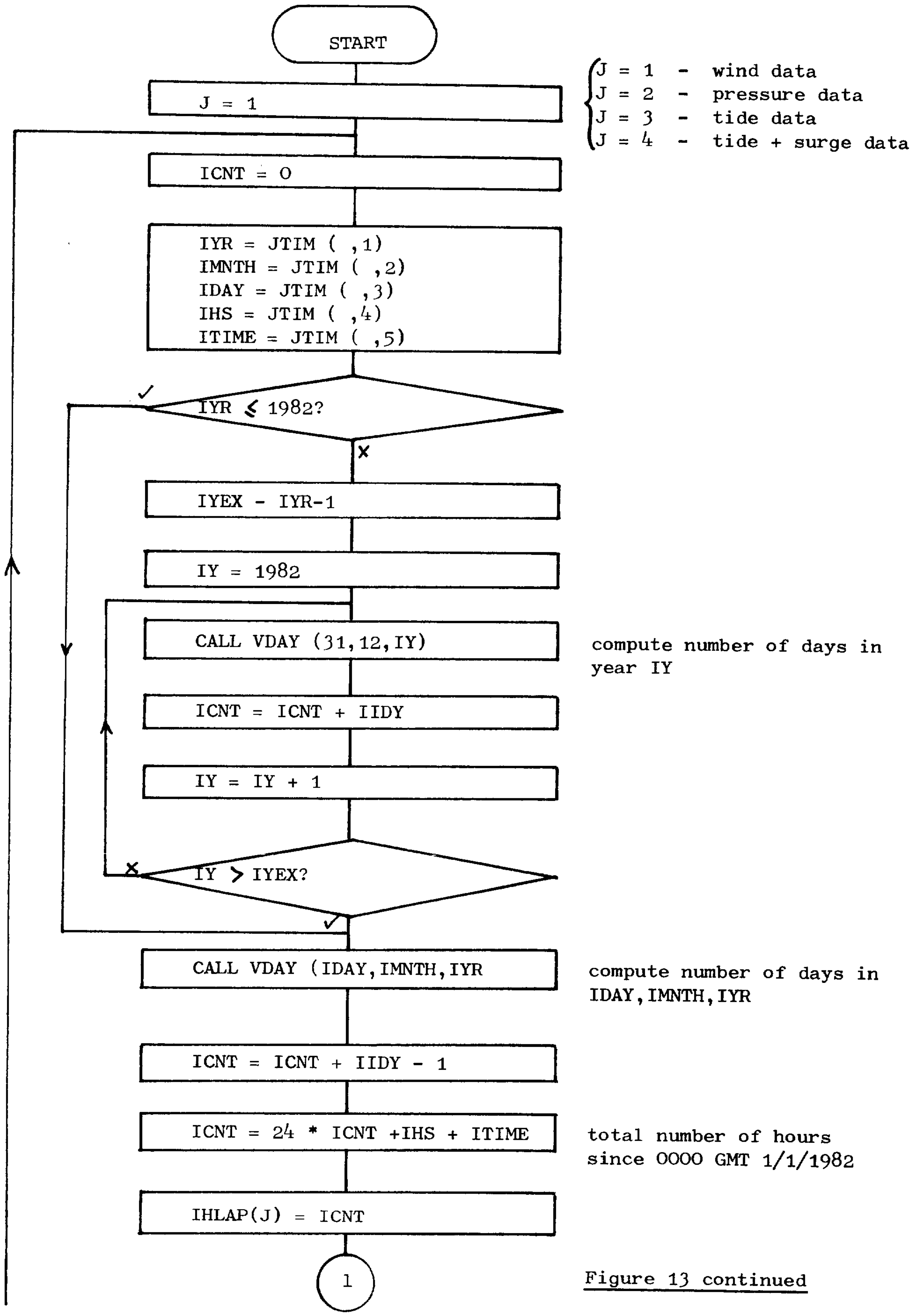


Figure 13 continued

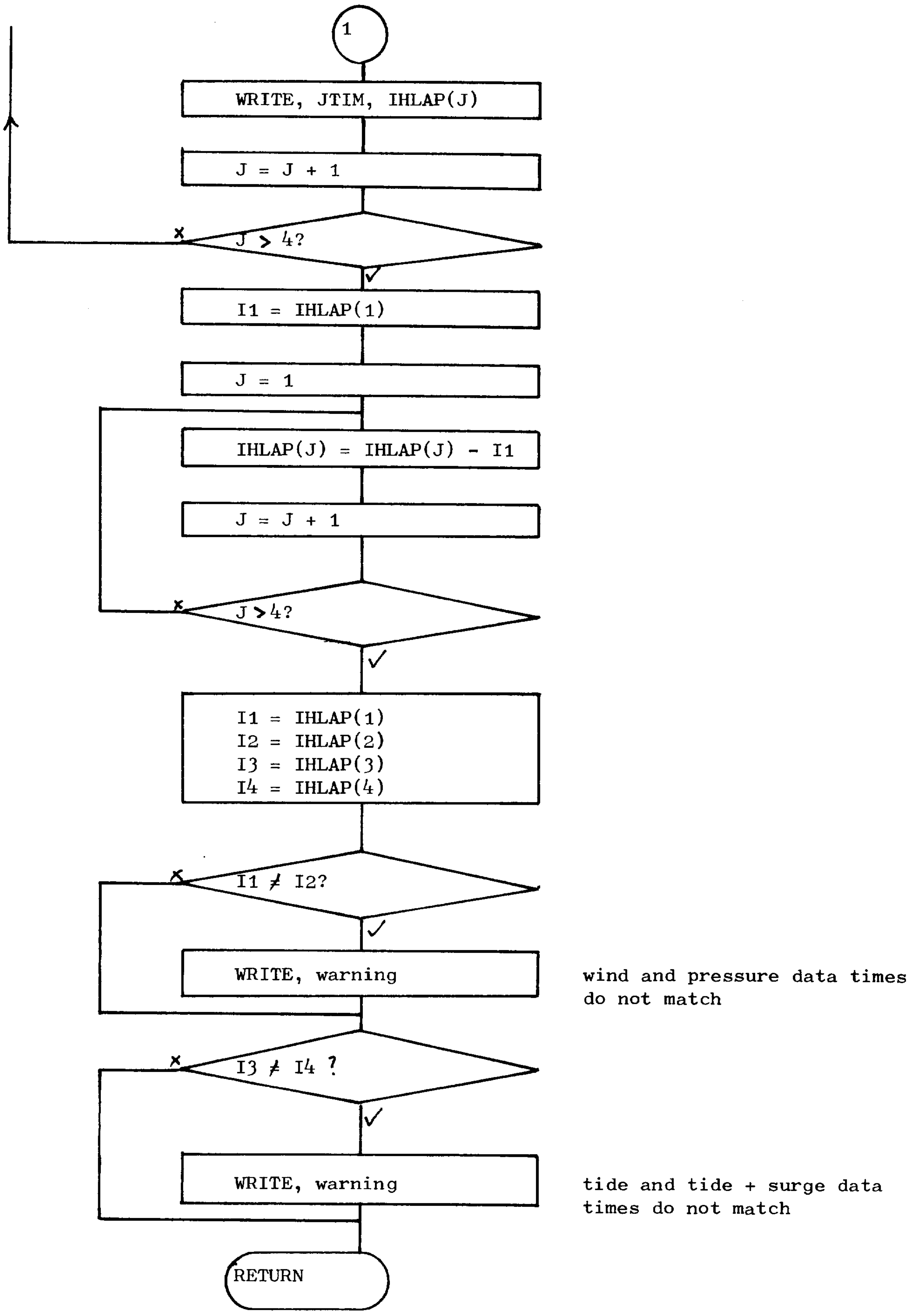
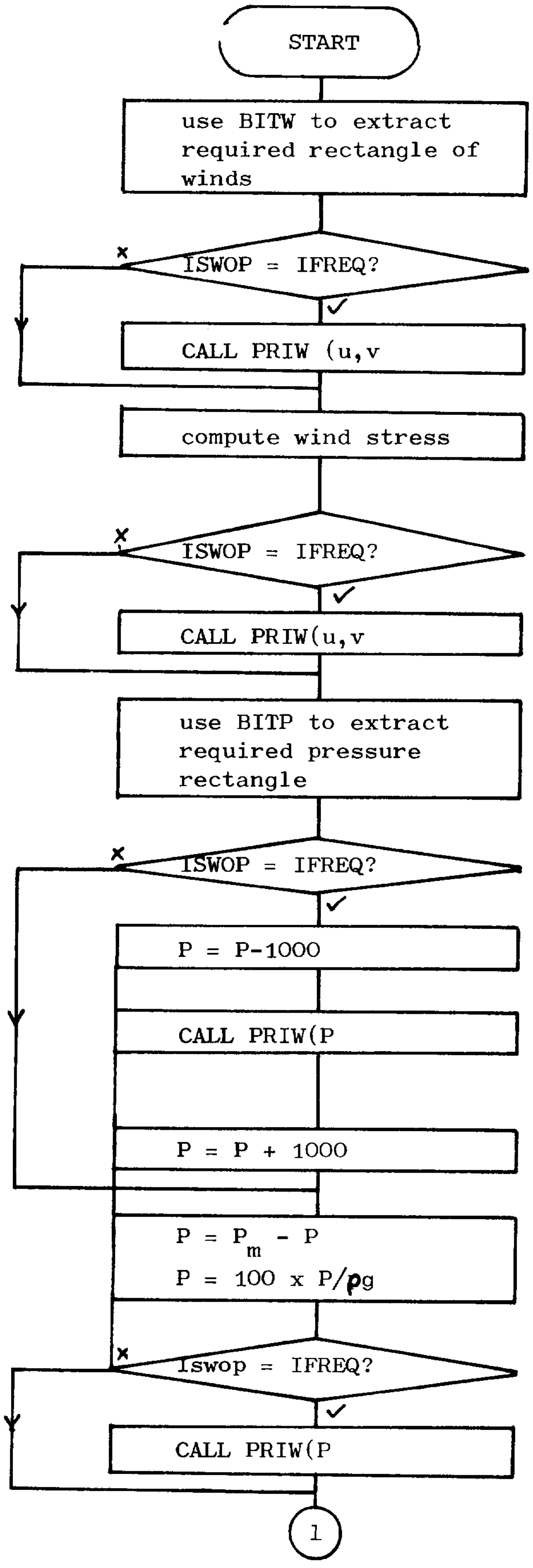


Figure 13 continued

Subroutine MET PROC - process the meteorological data



both u and v components at atmospheric model points

print wind components

using AO, A1 in drag coefficient

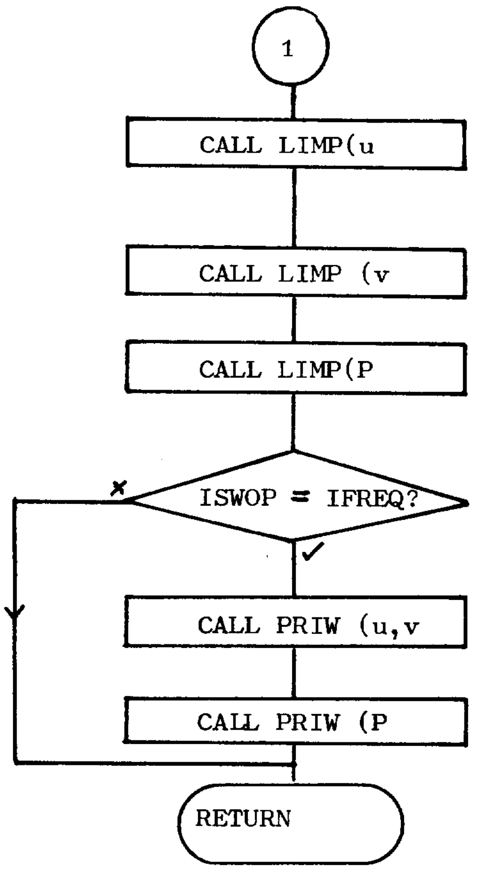
print strees components

print pressure deviations from 1000 mb

compute equivalent hydro-static sea surface elevation

print hydrostatic elevations

Figure 13 continued



linear interpolation of u-stresses to u-points of the sea model

v-stresses

hydrostatic elevations to ζ-points

print stresses and hydrostatic elevations at sea model points

Subroutine PRIW print either wind or pressure arrays

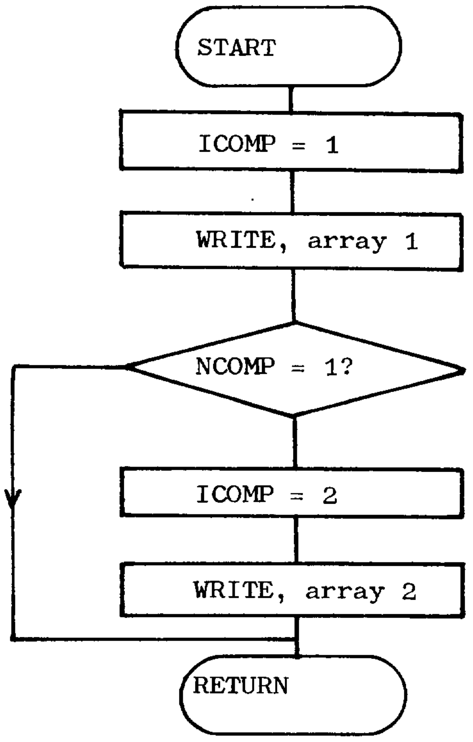


Figure 13 continued

Subroutine LIMP - linear interpolation of met data

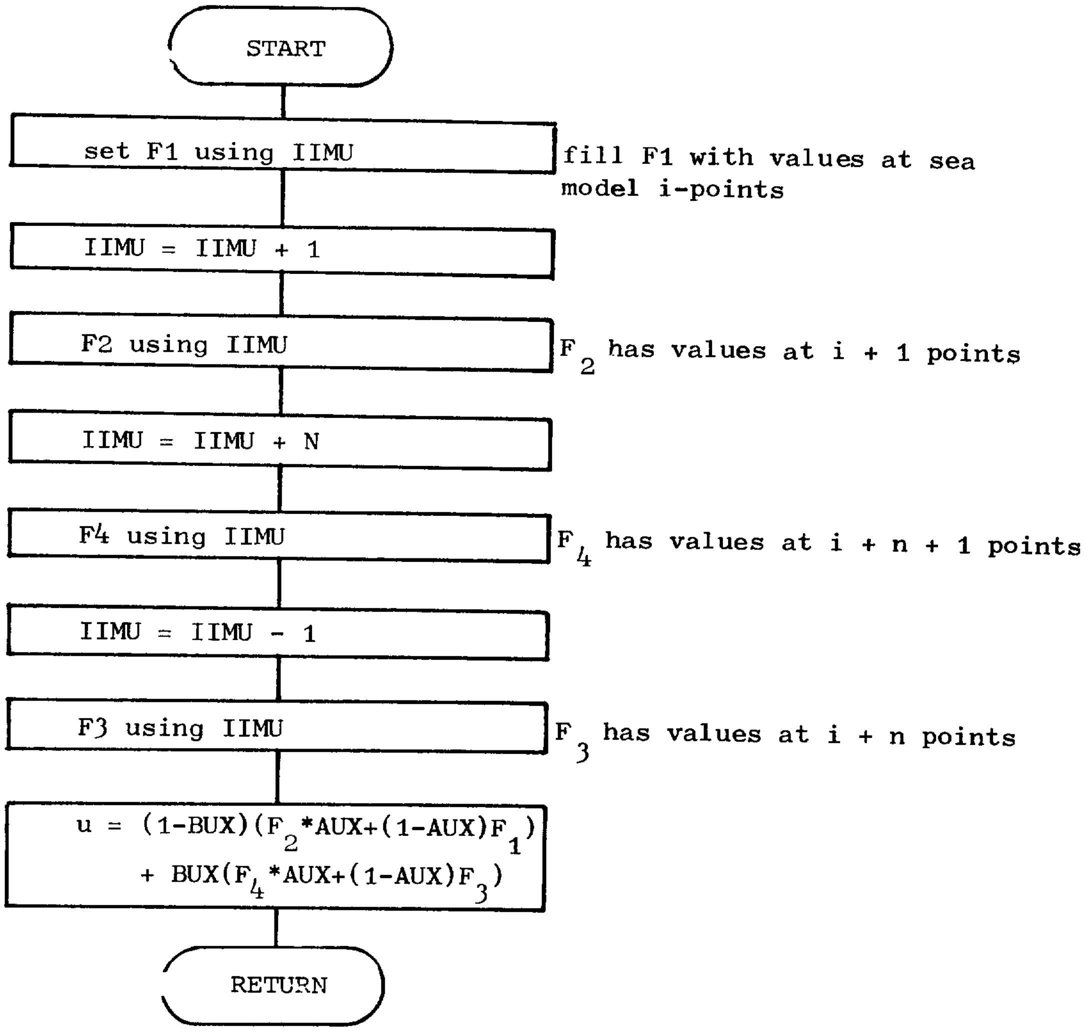
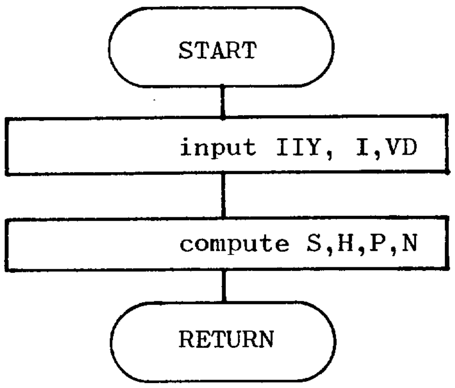

START
set F1 using IIMU
fill F1 with values at sea model i-points
IIMU = IIMU + 1
F2 using IIMU
F_2 has values at i + 1 points
IIMU = IIMU + N
F4 using IIMU
F_4 has values at i + n + 1 points
IIMU = IIMU - 1
F3 using IIMU
F_3 has values at i + n points
$u = (1\text{-BUX})(F_2*\text{AUX}+(1\text{-AUX})F_1) + \text{BUX}(F_4*\text{AUX}+(1\text{-AUX})F_3)$
RETURN

Figure 13 continued

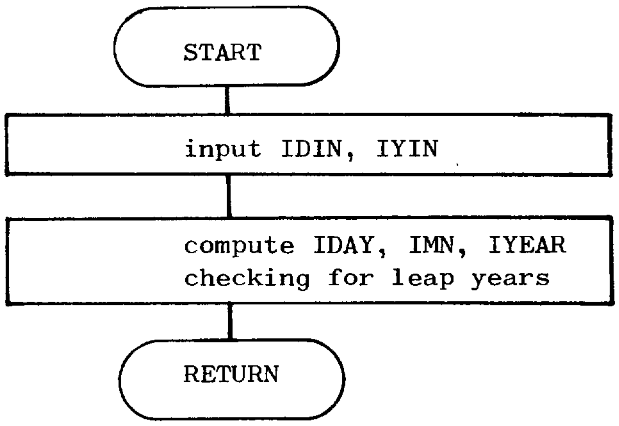
Subroutine SHPN - evaluate S,H,P, and N



year and day number

formula from Doodson (19

Subroutine DATEB - compute day, month, year given day number



Subroutine VDAY - computes day number in year given the date

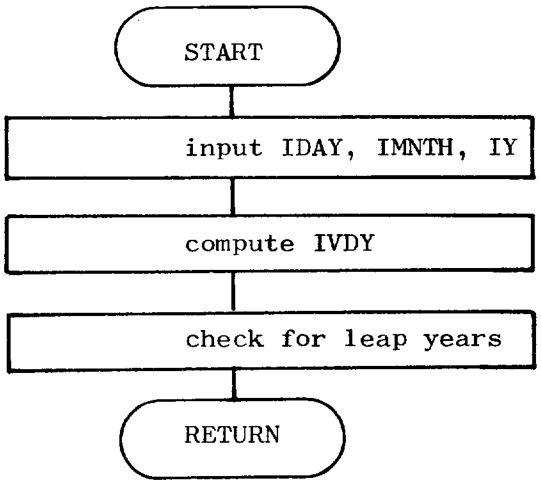
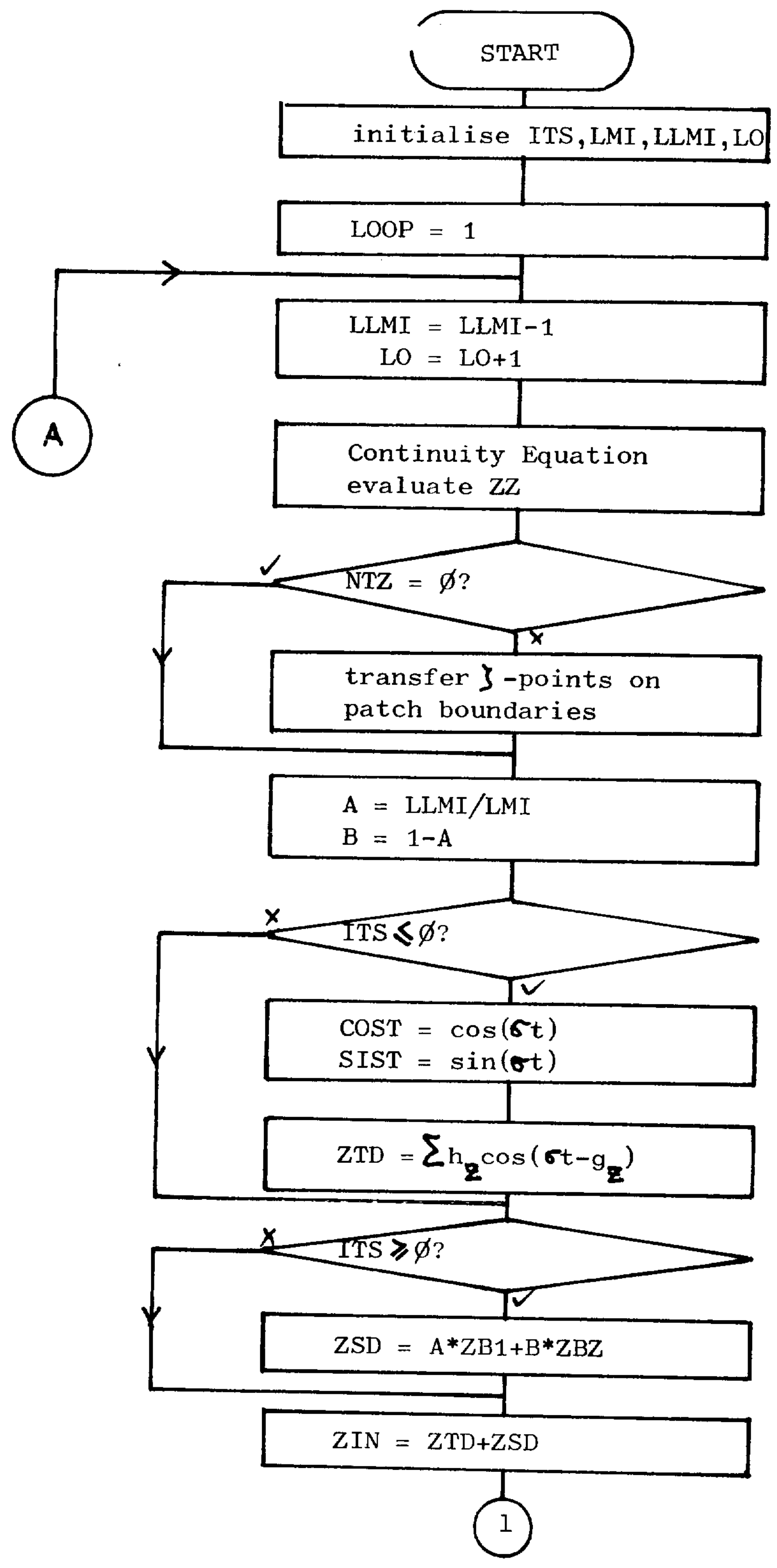


Figure 13 continued

Subroutine GESMEQ - integration of equations of continuity and motion



set up control parameters

begin timestepping

stove depths at -points in TBU,TBV for later use

any patches?

done through NPIN, NPOT

time interpolation coefficients

boundary tidal input

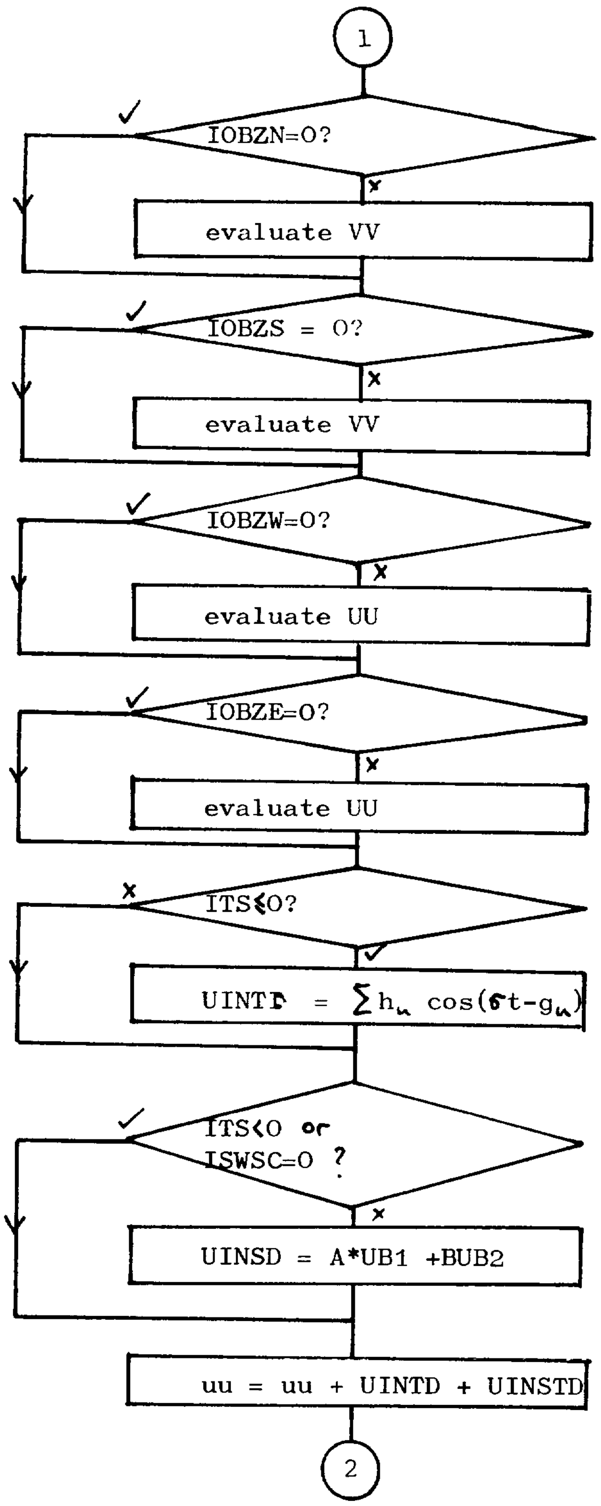
time dependent variation

tidal input

boundary surge input?

total prescribed boundary elevation

Figure 13 continued



RESPONSE CURRENT

northern boundary current

southern boundary current

western boundary current

eastern boundary current

Prescribed boundary tidal u-current

Prescribed boundary surge u-current

total boundary u-current

Figure 13 continued

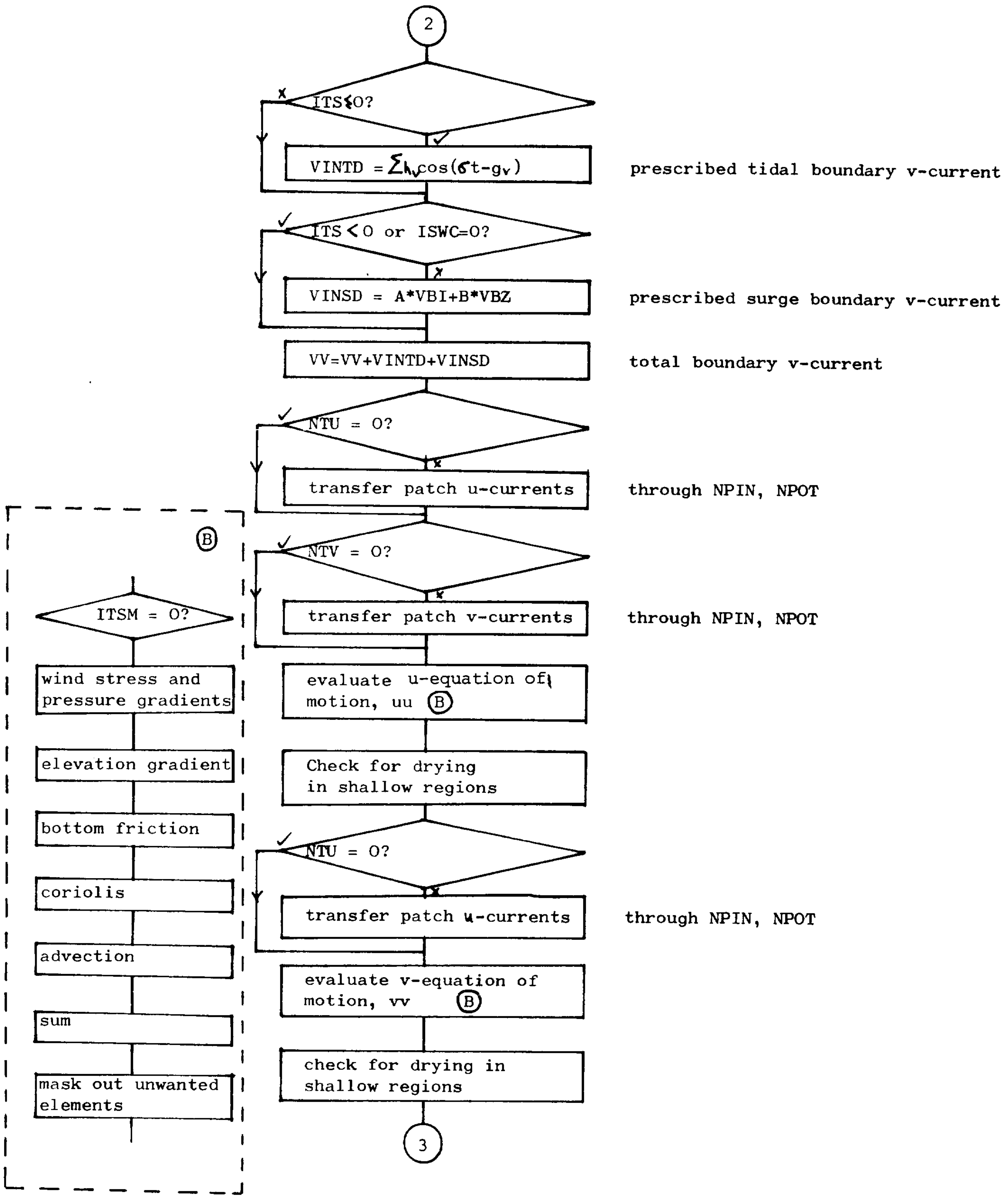
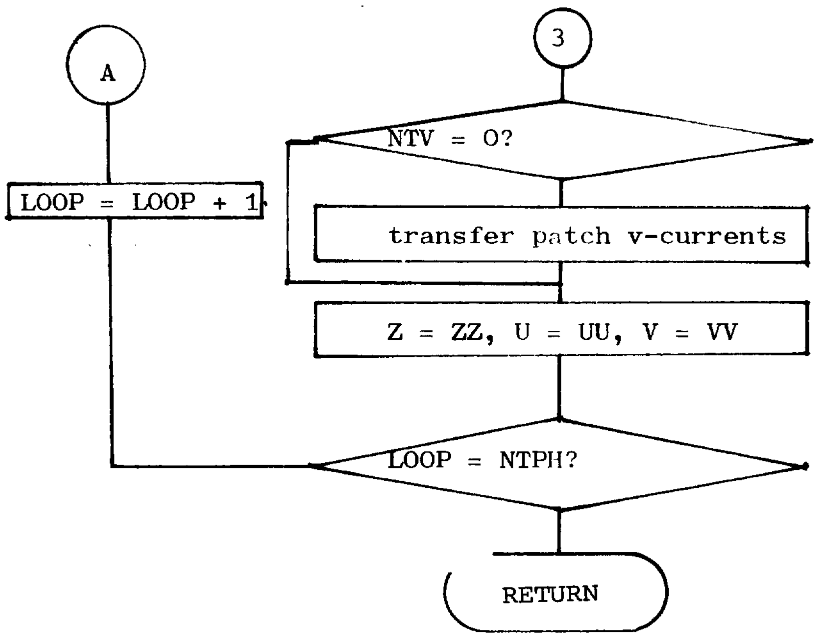

2
ITS≮0?
✗
✓
VINTD = $\Sigma h_v \cos(\sigma t - g_v)$
prescribed tidal boundary v-current
ITS<0 or ISWC=0?
✓
✗
VINSD = A*VBI+B*VBZ
prescribed surge boundary v-current
VV=VV+VINTD+VINSD
total boundary v-current
NTU = 0?
✓
✗
transfer patch u-currents
through NPIN, NPOT
NTV = 0?
✓
✗
transfer patch v-currents
through NPIN, NPOT
evaluate u-equation of motion, uu B
Check for drying in shallow regions
NTU = 0?
✓
✗
transfer patch u-currents
through NPIN, NPOT
evaluate v-equation of motion, vv B
check for drying in shallow regions
3
B
ITSM = 0?
wind stress and pressure gradients
elevation gradient
bottom friction
coriolis
advection
sum
mask out unwanted elements

Figure 13 continued



through NPIN, NPOT

move new arrays to old arrays for next timestep

end of integrations?

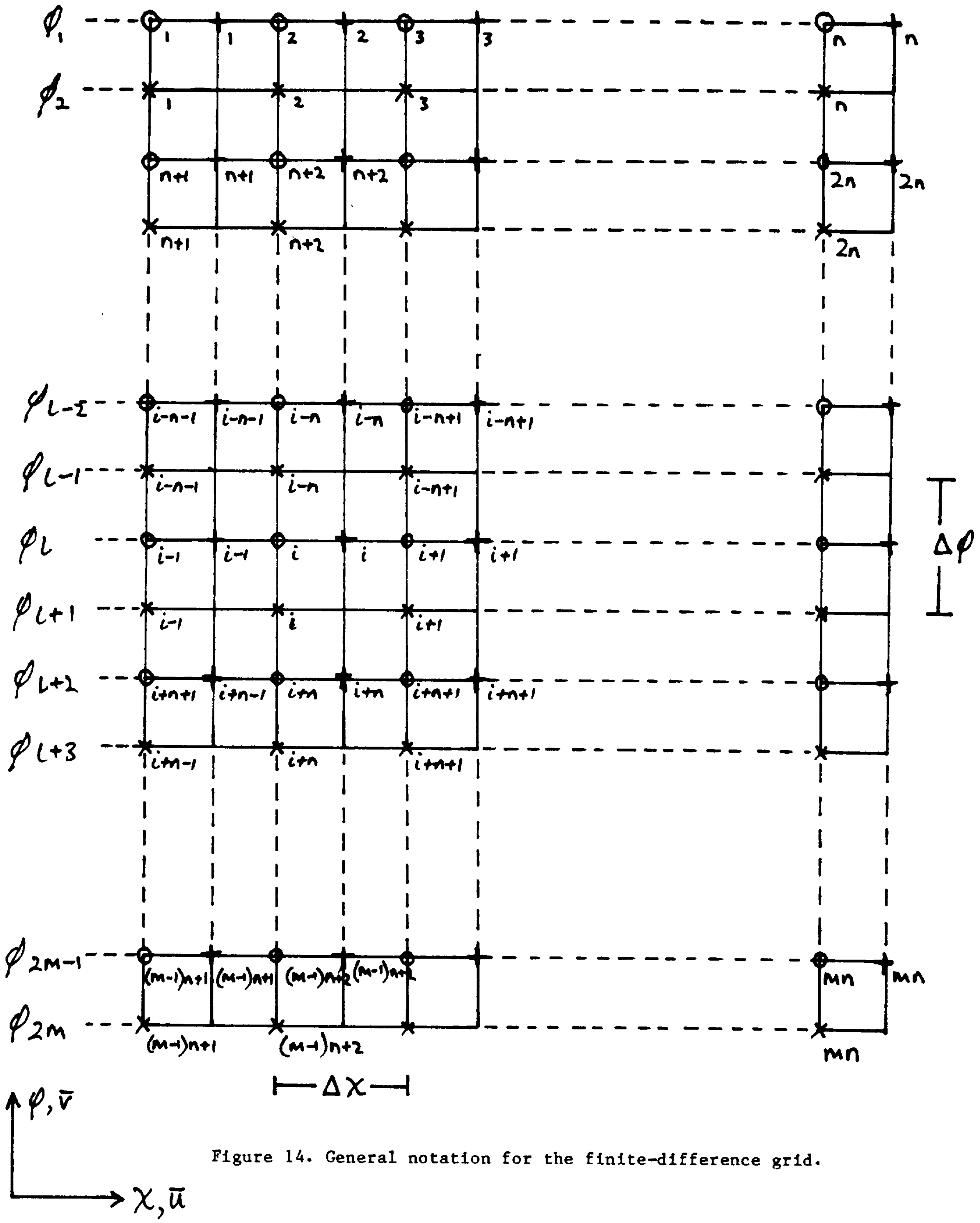


Figure 14. General notation for the finite-difference grid.

APPENDIX I

Finite difference equations

The difference grid scheme used in the model is shown in Figure 14. This consists of a rectangular array of m rows and n columns, with a grid spacing of $\Delta\varphi$ in the south-north direction and $\Delta\chi$ in the west-east direction. Elements, each consisting of a ζ-point denoted by o, a $\bar{u}$-point denoted by + and a $\bar{v}$-point denoted by x, are numbered consecutively

$i=1,\ldots,n,n+1,\ldots,2n,\ldots(m-1)n+1,\ldots mn,$

counting by element from left to right along each row, and moving down row by row.

Discrete values of the variables at appropriate grid points are identified by subscripts as follows

$\zeta = \zeta_i$, $h = h_i$ at ζ-point i,

$\bar{u} = \bar{u}_i$ at $\bar{u}$-point i,

$\bar{v} = \bar{v}_i$ at $\bar{v}$-point i.

The basic equations (1) to (3) are represented in finite difference form as follows:

continuity

$$\frac{\zeta_i^{t+\Delta t} - \zeta_i^t}{\Delta t} + \frac{1}{R\cos\varphi_i}\left\{\frac{d_i^t\bar{u}_i^t - d_{i-1}^t\bar{u}_{i-1}^t}{\Delta\chi} + \frac{e_{i-n}^t\bar{v}_{i-n}^t\cos\varphi_{i-1} - e_i^t\bar{v}_i^t\cos\varphi_{i+1}}{\Delta\varphi}\right\} = 0$$

where Δt is the timestep

$$d_i^t = 0.5(h_i + \zeta_i^t + h_{i+1} + \zeta_{i+1}^t)$$

$$e_i^t = 0.5(h_i + \zeta_i^t + h_{i+n} + \zeta_{i+n}^t)$$

u-equation of motion

$$\frac{\bar{u}_i^{t+\Delta t} - \bar{u}_i^t}{\Delta t} + \frac{1}{R\cos\varphi_i}\frac{1}{2}\left(\frac{\bar{u}_{i+1}^{t\,2} - \bar{u}_{i-1}^{t\,2}}{2\Delta\chi}\right) - 2\omega\sin\varphi_i\,\tilde{\bar{v}}_i^t$$

$$+ \frac{1}{2R}\left\{\frac{(\bar{v}_{i-n}^t + \bar{v}_{i-n+1}^t)}{2}\frac{(\bar{u}_{i-n}^t - \bar{u}_i^t)}{\Delta\varphi} + \frac{(\bar{v}_i^t + \bar{v}_{i+1}^t)}{2}\frac{(\bar{u}_i^t - \bar{u}_{i+n}^t)}{\Delta\varphi}\right\}$$

$$= \frac{g}{R\cos\varphi_i}\left\{\frac{\zeta_{i+1}^{t+\Delta t} - \zeta_i^{t+\Delta t}}{\Delta\chi}\right\} - \frac{k\,\bar{u}_i^{t+\Delta t}(\bar{u}_i^{t\,2} + \tilde{\bar{v}}_i^{t\,2})^{1/2}}{d_i^t} + \frac{1}{\rho}\left\{-P_i^t + \frac{F_i^t}{d_i^t}\right\}$$

v-equation of motion

$$\frac{\bar{v}_i^{t+\Delta t} - \bar{v}_i^t}{\Delta t} + \frac{1}{R\cos\phi_{i+1}} \frac{1}{2}\left\{ \frac{(\bar{u}_i^t + \bar{u}_{i+n}^t)}{2} \frac{(\bar{v}_{i+1}^t - \bar{v}_i^t)}{\Delta\chi} + \frac{(\bar{u}_{i-1}^t + \bar{u}_{i+n-1}^t)}{2} \frac{(\bar{v}_i^t - \bar{v}_{i-1}^t)}{\Delta\chi} \right\}$$

$$+ \frac{1}{2R}\left(\frac{\bar{v}_{i-n}^{t^2} - \bar{v}_{i+n}^{t^2}}{2\Delta\phi} \right) + 2\omega\sin\phi_{i+1}\,\tilde{\bar{u}}_i^{t+\Delta t}$$

$$= \frac{g}{R}\left\{ \frac{\zeta_i^{t+\Delta t} - \zeta_{i+n}^{t+\Delta t}}{\Delta\phi} \right\} - \frac{K\,\bar{v}_i^{t+\Delta t}\left(\tilde{\bar{u}}_i^{t^2} + \bar{v}_i^{t^2}\right)^{1/2}}{e_i^t} + \frac{1}{\rho}\left\{ -Q_i^t + \frac{G_i^t}{e_i^t} \right\}$$

where $\tilde{\bar{u}}_i^t = 0.25(\bar{u}_i^t + \bar{u}_{i-1}^t + \bar{u}_{i+n-1}^t + \bar{u}_{i+n}^t)$

$\tilde{\bar{v}}_i^t = 0.25(\bar{v}_i^t + \bar{v}_{i+1}^t + \bar{v}_{i-n+1}^t + \bar{v}_{i-n}^t)$

$P_i = \frac{1}{R\cos\phi} \frac{\partial p_a}{\partial\chi}$ at $\bar{u}$-point i

$Q_i = \frac{1}{R} \frac{\partial p_a}{\partial\phi}$ at $\bar{v}$-point i

$F_i = F_s$ at $\bar{u}$-point i

$G_i = G_s$ at $\bar{v}$-point i

and $\frac{\bar{v}}{R\cos\phi}\frac{\partial(\bar{u}\cos\phi)}{\partial\phi}$ is approximated by $\frac{\bar{v}}{R}\frac{\partial\bar{u}}{\partial\phi}$ in the $\bar{u}$-equation and the term $\frac{\bar{u}^2\tan\phi}{R}$ is ignored in the $\bar{v}$-equation.

Unlike the old scheme, which used an 'angled derivative' representation of the advective terms, the new system evaluates these terms uniformly at the lower time level, t. Although this is, in principle, less satisfactory it does make complete vectorisation possible. The scheme appears to run satisfactorily, integrating on occasions for over 80,000 cycles.

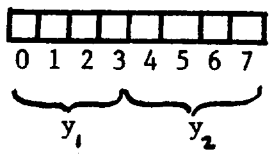
APPENDIX II

G-Bit explanation

In CYBER 200 Fortran special call statements there is an 8-bit designator (G-Bit) which allows various operations to be performed on the result. Consider any special call i.e.

CALL $Q8_{xxxx}$(X'yy',,AD,,BD,BITD,CD)

the G-Bit is X'y_1 y_2' where y_1 y_2 are two words, y_1 covering bits 0-3 and y_2 covering bits 4-7, thus



then individual bits can be switched '0' or '1' to describe the following operations:

bit	set to	means
0	0	64-bit operands
	1	32-bit operands
1	0	control store on ones
	1	control store on zero's
2	0	no offset for result
	1	result is offset, do not use
3	0	AD is a descriptor
	1	broadcast a constant from AD
4	0	BD is a descriptor
	1	broadcast a constant from BD
5,6	00	take A as is
	01	-A
	10	abs(A)
	11	-abs(A)

7	0	take B as is
	1	abs(B)

Therefore, if the G=Bit has value X'00', all bits are zero and this means,

64-bit operands

control store on ones i.e. $A_n \rightarrow C_n$ if BIT_n = 1 and

$B_n \rightarrow C_n$ if BIT_n = 0

no offset

AD is a descriptor

BD is a descriptor

take A as is

take B as is

If the G-Bit has value X'1515', all bits are one i.e. y_1 =1111 = 15 in binary code, y_2 = 1111 = 15

32-bit operands

control store on zero's i.e. $A_n \rightarrow C_n$ if BIT_n = 0 and

$B_n \rightarrow C_n$ if BIT_n = 1

offset

AD is a scalar

BD is a scalar

take A as -abs(A)

take B as abs(B)

APPENDIX III - Listings of computer programs

```
      PROGRAM SETUP(UNIT5=INPUT,UNIT6=OUTPUT,UNIT12=CPSMD3)
C**********************************************************************
C*                                                                    *
C*    PROGRAM SETUP                                                   *
C*    SETS UP SEA MODEL DATA IN A FILE                                *
C*    CYBER VERSION 16/08/82                                          *
C*                                                                    *
C**********************************************************************
      DIMENSION H(3072),LAB(3072),IZOB(150),LABZ(150),IUOB(150),
     C      IVOB(150),IZOBN(150),IZOBS(150),IZOBW(150),IZOBE(150),
     C       CSPUAI(3072),CSPVA(3072),CORUA(3072),CORVA(3072)
      DIMENSION HX(3072),LABX(3072),CSPU(3072),CSPV(3072),CORU(3072),
     C  CORV(3072)
      DIMENSION IXST(10),JXST(10),NCT(10),NRT(10),IF(10),JF(10),
     C     IXSTO(20),JXSTO(20),NCTO(20),NRTO(20)
      DIMENSION NPIN(3,100),NPOT(3,100)
      BIT BITZ,BITU,BITV,BITZD,BITUD,BITVD
      BIT BITUX,BITVX
      BIT BITIO
      BIT BITZO,BITZOD
      DIMENSION ID(3072)
      DIMENSION BITZ(3072),BITU(3072),BITV(3072),BITUX(3072),BITVX(3072)
      DIMENSION BITZO(3072)
      DIMENSION IDD(3072)
      COMMON/BUFIT/BITIO(32960)
      EQUIVALENCE (BITZ(1),BITIO(1)),(BITU(1),BITIO(3073)),
     C            (BITV(1),BITIO(6145)),(BITUX(1),BITIO(9217)),
     C            (BITVX(1),BITIO(12289)),(BITZO(1),BITIO(15361))
      COMMON/BUFRI/DATA(17408)
      EQUIVALENCE (DATA(1),H(1)),(DATA(3073),CORUA(1)),
     C  (DATA(6145),CORVA(1)),(DATA(9217),CSPUAI(1)),
     C  (DATA(12289),CSPVA(1)),(DATA(15361),IZOB(1)),
     C  (DATA(15511),IUOB(1)),(DATA(15661),IVOB(1)),
     C  (DATA(15811),IZOBN(1)),(DATA(15961),IZOBS(1)),
     C  (DATA(16111),IZOBW(1)),(DATA(16261),IZOBE(1)),
     C  (NRR,DATA(16500)),(NCC,DATA(16501)),(ITOT,DATA(16502)),
     C  (IINZ,DATA(16503)),(IMEU,DATA(16504)),(IMEV,DATA(16505)),
     C  (IINUX,DATA(16506)),(IINVX,DATA(16507)),(IOBZ,DATA(16508)),
     C  (IOBU,DATA(16509)),(IOBV,DATA(16510)),(ICN,DATA(16511)),
     C  (ICS,DATA(16512)),(ICW,DATA(16513)),(ICE,DATA(16514)),
     C  (DX,DATA(16515)),(DY,DATA(16516))
      EQUIVALENCE (NTZ,DATA(16517)),(NTU,DATA(16518)),
     C   (NTV,DATA(16519)),(NPIN(1,1),DATA(16520)),
     C   (NPOT(1,1),DATA(16820))
      DESCRIPTOR BITZD,BITUD,BITVD,ISEAD
      DESCRIPTOR LABD
      DESCRIPTOR BITZOD
      DESCRIPTOR ISEDD
C
C     THIS PROGRAM CALCULATES ALL LIMITS FOR OPERATIONAL SURGE
C     PREDICTION PROGRAMS
C     CYBER VERSION  16/08/82
C
      IREAD=5
      IRITE=6
      IPUN=7
      NRS=12
      READ(IREAD,101) NRX,NCX
      WRITE(IRITE,102) NRX,NCX
  101 FORMAT(20I4)
  102 FORMAT(1H1////' NUMBER OF ROWS, NRX=',I4//' NUMBER OF COLUMNS, NCX
     C=',I4)
      READ(IREAD,101) NRR,NCC,IFX,JFX,NPATCH
      WRITE(IRITE,99) NRR,NCC,IFX,JFX,NPATCH
   99 FORMAT(//' NUMBER OF ROWS IN NEW ARRAY, NRR=',I4//' NUMBER·OF COLU
     CMNS IN NEW ARRAY, NCC=',I4//' SHIFT COLUMN, IFX=',I4//' SHIFT ROW,
     C JFX=',I4//' NUMBER OF PATCHES, NPATCH=',I4)
      IF(NPATCH.EQ.0) GOTO 98
C
C     ALL POINT MOVES PREFORMED IN LARGE RECTANGLE
C
C
C     READ DATA FOR PATCH RELOCATION
C
      DO 35 I=1,NPATCH
      READ(IREAD,101) IXST(I),JXST(I),NCT(I),NRT(I),IF(I),JF(I)
   35 CONTINUE
C
C     READ DATA FOR REMOVAL OF OVERLAP LABELS
C
      NPAT2=2*NPATCH
      DO 17 I=1,NPAT2
   17 READ(IREAD,101) IXSTO(I),JXSTO(I),NCTO(I),NRTO(I)
C
C     READ POINTS, IN PATCHED MODEL COORDS, FOR DATA TRANSFER
C     BETWEEN PATCHES
C     NPIN(J,...  INPUT POINT FOR TRANSFER OF Z (J=1), U (J=2), V(J=3)
C     NPOT(J,... OUTPUT POINT FOR TRANSFER OF Z (J=1), U (J=2), V(J=3)
C
      READ(IREAD,101) NTZ,NTU,NTV
      READ(IREAD,101) ((NPIN(1,I),NPOT(1,I)),I=1,NTZ)
      READ(IREAD,101) ((NPIN(2,I),NPOT(2,I)),I=1,NTU)
      READ(IREAD,101) ((NPIN(3,I),NPOT(3,I)),I=1,NTV)
C
   98 CONTINUE
      ITOTX=NRX*NCX
C     TOTAL NUMBER OF POINTS
      ILAT=2*NRX
C     NUMBER OF LATITUDES COVERED
      READ(IREAD,103) DX,DY
      WRITE(IRITE,104) DX,DY
  103 FORMAT(2F12.9)
  104 FORMAT(1H0,34H E-W GRID INCREMENT IN RADIANS DX=,F12.9//35H  N-S G
     CRID INCREMENT IN RADIANS DY=,F12.9)
      READ(IREAD,114) PHIN
  114 FORMAT(F10.5)
      WRITE(IRITE,105) PHIN
  105 FORMAT(1H0,53H LATITUDE OF NORTHERNMOST ELEVATION POINTS IN DEGREE
     CS,F10.5)
      DCONV=3.1415926535/180.0
      PHIN=PHIN*DCONV
      READ(IREAD,101) (LABX(I),I=1,ITOTX)
      READ(IREAD,110) (HX(I),I=1,ITOTX)
  110 FORMAT(10F8.2)
      WRITE(IRITE,106)
  106 FORMAT(1H1,8H  LABELS//)
```

```
      CALL ARPRIN(DUM,LABX,NCX,NRX,1)
      WRITE(IRITE,107)
  107 FORMAT(1H1,19H  DEPTHS IN METRES //)
      CALL ARPRIN(HX,IDUM,NCX,NRX,2)
      TW=(2.0*0.2625)/3600.0
C     TWICE EARTHS ANGULAR VELOCITY
      N=NCX
C
C     COMPUTE LATITUDE DEPENDENT TERMS
C
      DO 2 J=1,ILAT
      PHI=PHIN-FLOAT(J-1)*DY*0.5
      CRP=TW*SIN(PHI)
      CSP=COS(PHI)
      I1=((J-1)/2)*N+1
      I2=I1+N-1
      IF(J.EQ.(J/2)*2) GOTO 4
      DO 3 I=I1,I2
      CSPU(I)=1.0/CSP
    3 CORU(I)=CRP
      GOTO 2
    4 DO 5 I=I1,I2
      CSPV(I)=CSP
    5 CORV(I)=CRP
    2 CONTINUE
      IF(NPATCH.NE.0) GOTO 97
      NTZ=0
      NTU=0
      NTV=0
      NRR=NRX
      NCC=NCX
      ITOT=NRR*NCC
      DO 18 I=1,ITOT
      H(I)=HX(I)
      LAB(I)=LABX(I)
      CSPUAI(I)=CSPU(I)
      CSPVA(I)=CSPV(I)
      CORUA(I)=CORU(I)
      CORVA(I)=CORV(I)
   18 CONTINUE
      GOTO 96
   97 CONTINUE
      ITOT=NRR*NCC
C
C     MAKE MODS TO ARRAYS LAB AND H FOR PATCHES
C
C     1. MAKE COPY OF LABX IN LAB FOR CONTROL OF PATCH TRANSFERS
C
      DO 301 I=1,ITOTX
  301 LAB(I)=LABX(I)
C
C
C     2. RELOCATE PATCHES
C
      DO 21 N=1,NPATCH
      NCTN=NCT(N)
      NRTN=NRT(N)
      DO 22 IC=1,NCTN
      DO 22 JR=1,NRTN
      IX=(JXST(N)+JR-2)*NCX+IXST(N)+IC-1
      IP=IX+JF(N)*NCX+IF(N)
      IF(LABX(IX).NE.0.AND.LABX(IP).NE.0) WRITE(NW,300) N,IC,JR
  300 FORMAT(///' DATA BEING OVERWRITTEN IN PATCH TRANSFER ',3I5)
      IF(LAB(IX).EQ.0) GOTO 23
      LABX(IP)=LABX(IX)
      HX(IP)=HX(IX)
      CSPU(IP)=CSPU(IX)
      CSPV(IP)=CSPV(IX)
      CORU(IP)=CORU(IX)
      CORV(IP)=CORV(IX)
   23 CONTINUE
   22 CONTINUE
   21 CONTINUE
C
C     3. RESTORE LAB TO ZERO
C
      DO 302 I=1,ITOTX
  302 LAB(I)=0
C
C     4. REMOVE OVERLAPS
C
      DO 25 N=1,NPAT2
      NPON=NCTO(N)+NRTO(N)
      IPS=(JXSTO(N)-1)*NCX+IXSTO(N)
      IP=IPS
      DO 24 IJ=1,NPON
      IF(NCTO(N).NE.0.AND.IJ.NE.1) IP=IP+1
      IF(NRTO(N).NE.0.AND.IJ.NE.1) IP=IP+NCX
      LABX(IP)=0
   24 CONTINUE
   25 CONTINUE
C
C     5. EXTRACT RECTANGLE
C
      DO 20 I=1,NCC
      DO 20 J=1,NRR
      II=(J-1)*NCC+I
      IX=(J+JFX-1)*NCX+I+IFX
      LAB(II)=LABX(IX)
      H(II)=HX(IX)
      CSPUAI(II)=CSPU(IX)
      CSPVA(II)=CSPV(IX)
      CORUA(II)=CORU(IX)
      CORVA(II)=CORV(IX)
   20 CONTINUE
      WRITE(IRITE,106)
      CALL ARPRIN(DUM,LAB,NCC,NRR,1)
      WRITE(IRITE,107)
      CALL ARPRIN(H,IDUM,NCC,NRR,2)
   96 CONTINUE
C
C     CALCULATE PARAMETERS TO BE DETERMINED FOR OPEN BOUNDARY POINTS
C
      IINZ=0
      IOBZ=0
      IOBU=0
      IOBV=0
      DO 10 I=1,ITOT
      L=LAB(I)
      IF(L.GE.100) IINZ=IINZ+1
      IF(L.LT.200) GOTO 10
      IOBZ=IOBZ+1
      IZOB(IOBZ)=I
      LLN=LAB(I-NCC)
      LLS=LAB(I+NCC)
      LLW=LAB(I-1)
      LLE=LAB(I+1)
      IF(LLN.LT.0.AND.LLS.GE.0.AND.LLW.GE.0.AND.LLE.GE.0) LABZ(IOBZ)=1
      IF(LLN.GE.0.AND.LLS.LT.0.AND.LLW.GE.0.AND.LLE.GE.0) LABZ(IOBZ)=2
      IF(LLN.GE.0.AND.LLS.GE.0.AND.LLW.LT.0.AND.LLE.GE.0) LABZ(IOBZ)=3
      IF(LLN.GE.0.AND.LLS.GE.0.AND.LLW.GE.0.AND.LLE.LT.0) LABZ(IOBZ)=4
      IF(LLN.LT.0.AND.LLS.GE.0.AND.LLW.LT.0.AND.LLE.GE.0) LABZ(IOBZ)=5
      IF(LLN.GE.0.AND.LLS.LT.0.AND.LLW.LT.0.AND.LLE.GE.0) LABZ(IOBZ)=6
      IF(LLN.LT.0.AND.LLS.GE.0.AND.LLW.GE.0.AND.LLE.LT.0) LABZ(IOBZ)=7
      IF(LLN.GE.0.AND.LLS.LT.0.AND.LLW.GE.0.AND.LLE.LT.0) LABZ(IOBZ)=8
      L=L-200
      LAB(I)=LAB(I)-100
      IF(L.NE.20.AND.L.NE.21) GOTO 13
      LAB(I)=LAB(I)-10
   13 CONTINUE
      IF(L.NE.12.AND.L.NE.2) GOTO 14
      LAB(I)=LAB(I)-1
   14 CONTINUE
      L=LABZ(IOBZ)
      IF(L.LT.3) GOTO 15
      IOBU=IOBU+1
      IF(L.EQ.3.OR.L.EQ.5.OR.L.EQ.6) IUOB(IOBU)=I-1
      IF(L.EQ.4.OR.L.EQ.7.OR.L.EQ.8) IUOB(IOBU)=I
   15 CONTINUE
      IF(L.EQ.3.OR.L.EQ.4) GOTO 16
      IOBV=IOBV+1
      IF(L.EQ.1.OR.L.EQ.5.OR.L.EQ.7) IVOB(IOBV)=I-NCC
      IF(L.EQ.2.OR.L.EQ.6.OR.L.EQ.8) IVOB(IOBV)=I
   16 CONTINUE
   10 CONTINUE
C
C     COMPUTE N,S,W,E BOUNDARY POINTS
C
      ICN=0
      ICS=0
      ICW=0
      ICE=0
      DO 51 I=1,IOBZ
      IF(LABZ(I).EQ.1.OR.LABZ(I).EQ.5.OR.LABZ(I).EQ.7) GOTO 52
      GOTO 53
   52 ICN=ICN+1
      IZOBN(ICN)=I
   53 IF(LABZ(I).EQ.2.OR.LABZ(I).EQ.6.OR.LABZ(I).EQ.8) GOTO 54
      GOTO 55
   54 ICS=ICS+1
      IZOBS(ICS)=I
   55 IF(LABZ(I).EQ.3.OR.LABZ(I).EQ.5.OR.LABZ(I).EQ.6) GOTO 56
      GOTO 57
   56 ICW=ICW+1
```

```
      IZOBW(ICW)=I
   57 IF(LABZ(I).EQ.4.OR.LABZ(I).EQ.7.OR.LABZ(I).EQ.8) GOTO 58
      GOTO 51
   58 ICE=ICE+1
      IZOBE(ICE)=I
   51 CONTINUE
C
C     SET UP Z-MASK
C
      ASSIGN BITZD,BITZ(1;ITOT)
      ASSIGN LABD,LAB(1;ITOT)
      BITZD=LABD.GE.100
      IINZ=Q8SCNT(BITZD)
      ASSIGN ISEAD,ID(1;ITOT)
      ISEAD=LABD-100
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITZD,LABD)
C
C     SET UP U-MASK
C
      ASSIGN BITUD,BITU(1;ITOT)
      BITUD=LABD.GE.10
      IMEU=Q8SCNT(BITUD)
      ISEAD=LABD-10
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITUD,LABD)
C
C     SET UP V-MASK
C
      ASSIGN BITVD,BITV(1;ITOT)
      BITVD=LABD.GE.1
      IMEV=Q8SCNT(BITVD)
C
C     SET UP EXTENDED U-MASK
C
      ISEAD=1
      LABD=0
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITUD,ISEAD)
      DO 712 J=1,IOBU
      I=IUOB(J)
  712 ID(I)=1
      ASSIGN BITUD,BITUX(1;ITOT)
      BITUD=ISEAD.EQ.1
      IINUX=Q8SCNT(BITUD)
C
C     SET UP EXTENDED V-MASK
C
      ISEAD=1
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITVD,ISEAD)
      DO 713 J=1,IOBV
      I=IVOB(J)
  713 ID(I)=1
      ASSIGN BITVD,BITVX(1;ITOT)
      BITVD=ISEAD.EQ.1
      IINVX=Q8SCNT(BITVD)
C
C     SET UP MASK FOR OPEN BOUNDARY Z-POINTS
C
      ISEAD=0
      DO 714 J=1,IOBZ
      I=IZOB(J)
  714 ID(I)=1
      ASSIGN BITZOD,BITZO(1;ITOT)
      BITZOD=ISEAD.EQ.1
C
C     WRITE TO OUTPUT FILE
C
      CALL Q7BUFOUT(NRS,DATA,34,'SMALL')
      CALL Q7WAIT(NRS,DATA,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(IRITE,207) ISTAT
  207 FORMAT(10X,' ABNORMAL END OF BUFOUT  STAT= ',I5)
      CALL Q7BUFOUT(NRS,BITIO,1,'SMALL')
      CALL Q7WAIT(NRS,BITIO,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(IRITE,207) ISTAT
      WRITE(IRITE,203)
  203 FORMAT(1H1,21H  DATA STORED IN NRS //)
      WRITE(IRITE,202) NRR,NCC,ITOT,IINZ,IMEU,IMEV,IINUX,IINVX,
     C                 IOBZ,IOBU,IOBV,ICN,ICS,ICW,ICE
      WRITE(IRITE,202) (IZOB(J),J=1,IOBZ)
      WRITE(IRITE,202) (IUOB(J),J=1,IOBU)
      WRITE(IRITE,202) (IVOB(J),J=1,IOBV)
      IF(ICN.NE.0) WRITE(IRITE,202) (IZOBN(I),I=1,ICN)
      IF(ICS.NE.0) WRITE(IRITE,202) (IZOBS(I),I=1,ICS)
      IF(ICW.NE.0) WRITE(IRITE,202) (IZOBW(I),I=1,ICW)
      IF(ICE.NE.0) WRITE(IRITE,202) (IZOBE(I),I=1,ICE)
      WRITE(IRITE,205) DX,DY
C     WRITE(IRITE,200) (H(I),I=1,ITOT)
      WRITE(IRITE,204) (CORUA(I),I=1,ITOT)
      WRITE(IRITE,204) (CORVA(I),I=1,ITOT)
      WRITE(IRITE,204) (CSPUAI(I),I=1,ITOT)
      WRITE(IRITE,204) (CSPVA(I),I=1,ITOT)
      ASSIGN ISEDD,IDD(1;ITOT)
      ISEAD=1
      LABD=0
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITZD,ISEAD)
      WRITE(IRITE,101) IINZ
      DO 90 J=1,NRR
      I1=(J-1)*NCC+1
      I2=J*NCC
      WRITE(IRITE,208) (ID(I),I=I1,I2)
   90 CONTINUE
  208 FORMAT(10X,100I1)
      ISEDD=0
      L=0
      DO 402 I=1,ITOT
      IF(ID(I).EQ.1) L=L+1
  402 IF(ID(I).EQ.1) IDD(I)=L
      WRITE(6,60)L
   60 FORMAT(1H1,' COMPACT Z, L=',I5)
      CALL PR(IDD,NCC,NRR)
      ISEAD=1
      ASSIGN BITUD,BITU(1;ITOT)
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITUD,ISEAD)
      WRITE(IRITE,101) IMEU
      DO 91 J=1,NRR
      I1=(J-1)*NCC+1
      I2=J*NCC
      WRITE(IRITE,208) (ID(I),I=I1,I2)
   91 CONTINUE
      ISEDD=0
      L=0
      DO 401 I=1,ITOT
      IF(ID(I).EQ.1) L=L+1
  401 IF(ID(I).EQ.1) IDD(I)=L
      WRITE(6,61)L
   61 FORMAT(1H1,' COMPACT U, L=',I5)
      CALL PR(IDD,NCC,NRR)
      ISEAD=1
      ASSIGN BITVD,BITV(1;ITOT)
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITVD,ISEAD)
      WRITE(IRITE,101) IMEV
      DO 92 J=1,NRR
      I1=(J-1)*NCC+1
      I2=J*NCC
      WRITE(IRITE,208) (ID(I),I=I1,I2)
   92 CONTINUE
      ISEDD=0
      L=0
      DO 403 I=1,ITOT
      IF(ID(I).EQ.1) L=L+1
  403 IF(ID(I).EQ.1) IDD(I)=L
      WRITE(6,62)L
   62 FORMAT(1H1,' COMPACT V, L=',I5)
      CALL PR(IDD,NCC,NRR)
      ISEAD=1
      ASSIGN BITUD,BITUX(1;ITOT)
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITUD,ISEAD)
      WRITE(IRITE,101) IINUX
      DO 93 J=1,NRR
      I1=(J-1)*NCC+1
      I2=J*NCC
      WRITE(IRITE,208) (ID(I),I=I1,I2)
   93 CONTINUE
      ISEAD=1
      ASSIGN BITVD,BITVX(1;ITOT)
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITVD,ISEAD)
      WRITE(IRITE,101) IINVX
      DO 94 J=1,NRR
      I1=(J-1)*NCC+1
      I2=J*NCC
      WRITE(IRITE,208) (ID(I),I=I1,I2)
   94 CONTINUE
      ISEAD=1
      ASSIGN BITZOD,BITZO(1;ITOT)
      CALL Q8MASKV(X'00',,ISEAD,,LABD,BITZOD,ISEAD)
      WRITE(IRITE,101) IOBZ
      DO 95 J=1,NRR
      I1=(J-1)*NCC+1
      I2=J*NCC
      WRITE(IRITE,208) (ID(I),I=I1,I2)
   95 CONTINUE
  200 FORMAT(12F10.2)
  202 FORMAT(2X,20I6)
  204 FORMAT(8E16.8)
  205 FORMAT(2X,2F12.9)
      STOP
      END
      SUBROUTINE ARPRIN(A,IA,NC,NR,IFO)
      DIMENSION A(1),IA(1)
      NW=6
      DO 10 IC=1,999
      IS=(IC-1)*20+1
      IE=IC*20
      IF(IE.GT.NC) IE=NC
      IF(IS.GT.NC) GOTO 12
      IF(IC.NE.1) WRITE(NW,100)
  100 FORMAT(1H1)
      DO 11 J=1,NR
      ISS=(J-1)*NC+IS
      IEE=(J-1)*NC+IE
      IF(IFO.EQ.1) WRITE(NW,101) J,(IA(II),II=ISS,IEE)
  101 FORMAT(1X,I5,5X,20I4)
      IF(IFO.EQ.2) WRITE(NW,102) J,(A(II),II=ISS,IEE)
  102 FORMAT(1X,I5,5X,20F5.0)
   11 CONTINUE
   10 CONTINUE
   12 CONTINUE
      RETURN
      END
      SUBROUTINE PR(ID,NC,NR)
      DIMENSION ID(1)
      DO 1 I=1,99
      IF(I.EQ.1) WRITE(6,10)
   10 FORMAT(///)
      IF(I.GT.1) WRITE(6,11)
   11 FORMAT(1H1)
      I1=(I-1)*25+1
      I2=I*25
      IF(I2.GT.NC) I2=NC
      IF(I1.GT.NC) RETURN
      DO 2 J=1,NR
      J1=(J-1)*NC+I1
      J2=(J-1)*NC+I2
      WRITE(6,12) J,(ID(JJ),JJ=J1,J2)
   12 FORMAT(1X,I2,2X,25I5)
    2 CONTINUE
    1 CONTINUE
      RETURN
      END
```

```
  52  57
  46  44   7   0   3                    Input data for program SETUP
   2  26   8  12  42 -25
  17  45  14   8  18  -6
  50  11   8  12  -6  18
   8  27   0  10
  51   1   0  12
  18  46  10   0
  35  39   8   0
  51  13   0   5
  44  32   0   4
  43  44  45
1146  881190 1321234 1761278 2201322 2641366 3081410 3521454 3961498 4401542 484
19471701194817021949170319501704195117051952170619531707 6151401 6591445 7031489
 7471533  871145 1311189 1751233 2191277 2631321 3071365 3511409 3951453 4391497
 48315411745199117461992174719931748199417491995175019961751199971358 5721402 616
1446 6601490 7041534 748
1146  881190 1321234 1761278 2201322 2641366 3081410 3521454 3961498 4401542 484
19471701194817021949170319501704195117051952170619531707 6151401 6591445 7031489
 7471533  871145 1311189 1751233 2191277 2631321 3071365 3511409 3951453 4391497
 48315411745199117461992174719931748199417491995175019961751199971358 5721402 616
1446 6601490 7041534 74817441990
1146  881190 1321234 1761278 2201322 2641366 3081410 3521454 3961498 4401542 484
19471701194817021949170319501704195117051952170619531707 6151401 6591445 7031489
 7471533  871145 1311189 1751233 2191277 2631321 3071365 3511409 3951453 4391497
 48315411745199117461992174719931748199417491995175019961751199971358 5721402 616
1446 6601490 7041534 7481102  44  431101
 0.008726646 0.005817764
  62.5
   0   0   0   0   0   0   0   0  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
  -1  -1  -1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0  -1 222 221 221 221 221 221 221 221 201   0 201 221 221 221 221
 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 200
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0  -1 212 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212
 111 111 111 111 111 111 111 111 101 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101 111 111 111 111 111
 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 101 110 111 111 111 111 111 111 111 111
 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 101   0 111 111 111 111 111 111 111 111 111 111 111
 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0  -1 212 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  -1 212 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111
```

```
111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 100 101 111 111 111
111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0
  0   0   0 101   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111
111 111 111 111 111 111 111 111 111 110 110 110 110 111 111 111 111 111 111 111
111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0 111 111
101   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111
111 111 100 111 111 100   0   0   0   0 111 111 111 111 111 111 111 111 111 111
111 111 111 111 111 111 111 101   0   0   0   0   0   0 111 111 111 111 101   0
  0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 101   0
111 101   0   0   0   0 111 111 111 111 111 111 111 111 111 111 111 111 111 111
111 111 111 111 111 101   0   0   0   0 111 111 111 111 111 101   0   0   0   0
  0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 100 111 110 100   0
  0   0 110 110 110 110 111 111 111 111 111 111 111 111 111 111 111 111 111 111
111 111 111 111 111 111 111 111 111 111 110 111 111 101   0   0   0   0   0   0
  0   0  -1 212 111 111 111 111 111 111 111 101   0 101   0   0   0   0   0   0
  0   0   0 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
111 111 111 111 111 111 100   0 111 111 101   0   0   0   0   0   0   0   0  -1
212 111 111 111 111 111 111 111 101 111 101   0   0   0   0   0   0   0   0   0
111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
110 110 100   0   0 111 111 111 101   0   0   0   0   0   0   0  -1 212 111 111
111 111 111 111 111 111 111 110 101   0   0   0   0   0   0   0 111 111 111 111
111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0
  0   0 110 111 111 101   0   0   0   0   0   0   0  -1 212 111 111 111 111 111
111 111 111 111 111 100   0   0   0   0   0   0 111 111 111 111 111 111 111 111
111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0
111 111 101   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111
111 111 110 101   0   0   0   0   0 110 110 111 111 111 111 111 111 111 111 111
111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0 111 111 110 100
  0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 111 101   0
101   0   0   0   0   0   0   0 110 111 111 111 111 111 111 111 111 111 111 111
111 111 111 111 111 111 111 101   0   0   0   0 110 111 100   0   0   0   0   0
  0   0   0   0  -1 212 111 111 111 111 111 111 111 110 110 110 110 101 111 100
  0   0   0   0   0   0 111 111 111 111 111 111 111 111 111 111 111 111 111 111
111 111 111 111 111 101   0   0   0   0 100   0   0   0   0   0   0   0   0   0
  0  -1 212 111 111 111 111 111 111 100   0   0   0   0 111 101   0   0   0   0
  0   0   0 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212
111 111 111 111 111 101   0   0   0   0   0 110 101   0 111 101   0   0   0   0
110 110 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111
110 110 100   0   0   0   0   0   0 111 111 111 101   0   0   0   0   0   0 111
111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0
  0   0   0   0   0   0   0   0  -1  -1  -1  -1  -1 211 111 111 101   0   0   0
  0   0   0   0   0 111 111 101 111 111 101   0   0   0   0   0 110 111 111 111
111 111 111 111 111 111 111 111 111 111 111 111 111 111 100   0   0   0   0   0
  0   0   0   0  -1 221 221 221 221 211 111 111 111 101   0   0   0   0   0   0
  0   0 111 111 111 111 111 101   0   0   0   0   0   0 111 111 111 111 111 111
111 111 111 111 111 111 111 111 110 110 100   0   0   0   0   0   0   0   0   0
 -1 212 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0 111
111 111 110 110 100   0   0   0   0   0   0 111 111 111 111 111 111 111 111 111
111 111 110 110 100   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111
111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0 111 111 101   0
  0   0   0   0   0   0   0   0 110 110 110 111 111 111 111 111 111 110 100   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111
111 111 111 111 111 101   0   0   0   0   0   0   0 111 111 111 101   0   0   0
  0   0   0   0   0   0   0   0 111 111 111 111 111 101   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111
111 111 100   0   0   0   0   0   0 111 111 111 111 100   0   0   0   0   0   0
```

```
  0   0   0   0   0 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 100   0
  0   0   0   0   0 111 111 111 111 100   0   0   0   0   0   0   0   0   0   0
  0   0 111 111 111 111 111 100   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 101   0   0   0   0
111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0 111 111
111 111 111 100   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0  -1 212 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
111 111 111 111 111 110 100   0   0   0   0   0   0   0 110 110 111 111 111 100
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1
210 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
111 100   0   0   0   0   0   0   0   0   0   0 111 110 110 100   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 220 220
220 220 220 210 111 111 111 111 111 111 111 111 111 111 111 111 111 100   0   0
  0   0   0   0   0   0   0   0 111 101   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1  -1  -1  -1  -1
 -1 212 111 111 111 111 111 111 111 111 111 111 111 100   0   0   0 111 111 111
111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111
111 111 111 111 111 111 111 111 111 101   0 111 111 111 111 111 111 111 111 111
111 111 110 100   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111
111 111 111 111 111 111 111 111 111 111 111 111 111 111 110 111 111 111 100   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111
111 111 111 111 111 111 111 111 111 111 101   0 110 110 100   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 111 111
111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0  -1 210 111 111 111 111 111 111 111 111 111 111 111 111 111
110 110 110 110 110 100   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0  -1 220 210 111 111 111 111 111 111 111 111 111 111 101   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0  -1  -1 220 220 220 210 111 111 111 111 111 111 101   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 -1  -1  -1  -1 210 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0  -1 220 220 210 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1
 -1  -1 210 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1
210 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 210 111
111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 210 111 111 111 111
111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0  -1 220 220 220 220 220 200
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0  -1  -1  -1  -1  -1  -1   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0
 2050.00 2020.00 1800.00 1550.00 1250.00 1120.00 1010.00  940.00  810.00  770.00
  670.00  600.00  580.00  560.00  440.00  320.00  160.00  130.00  140.00  110.00
  130.00  220.00  490.00  610.00  690.00  680.00  750.00 1500.00 1690.00 1620.00
 1410.00 1220.00 1010.00  780.00  830.00  620.00  500.00  440.00  330.00  200.00
  185.00  185.00  180.00   64.00    0.      0.      0.      0.      0.      0.  
    0.      0.      0.      0.      0.      0.      0.   2120.00 2080.00 1950.00
 1630.00 1410.00 1240.00 1070.00  990.00 1030.00 1010.00  905.00  830.00  690.00
  510.00  340.00  140.00   95.00    0.     60.00   70.00  170.00  180.00  260.00
  280.00  490.00  980.00 1620.00 1660.00 1620.00 1450.00  960.00  680.00  650.00
  560.00  440.00  390.00  390.00  400.00  350.00  225.00  115.00  150.00   73.00
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.  
    0.      0.      0.      0.   2290.00 2210.00 1960.00 1750.00 1540.00 1290.00
 1280.00 1290.00 1220.00 1140.00 1150.00  990.00  880.00  700.00  520.00  170.00
  130.00  100.00   70.00  110.00  220.00  190.00  270.00  540.00 1000.00 1490.00
 1570.00 1450.00  760.00  330.00  260.00  230.00  220.00  210.00  350.00  370.00
  410.00  385.00  385.00  270.00  200.00  140.00    0.      0.      0.      0.  
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.  
    0.   2340.00 2190.00 2010.00 1890.00 1780.00 1530.00 1450.00 1410.00 1250.00
 1190.00 1240.00 1230.00 1080.00  710.00  620.00  710.00  290.00  160.00   70.00
  180.00  180.00  200.00  360.00 1050.00 1230.00 1390.00 1430.00 1240.00  960.00
  320.00  180.00  185.00  180.00  185.00  170.00  210.00  280.00  380.00  360.00
  350.00  310.00  220.00    0.      0.      0.      0.      0.      0.      0.  
    0.      0.      0.      0.      0.      0.      0.      0.   2410.00 2200.00
 2030.00 1880.00 1710.00 1605.00 1400.00 1340.00 1270.00 1180.00 1110.00 1125.00
 1005.00  380.00  150.00  160.00  890.00  630.00  150.00  240.00  270.00  440.00
  980.00 1090.00 1130.00 1180.00 1090.00  670.00  420.00  160.00  150.00  160.00
  155.00  155.00  150.00  140.00  180.00  270.00  360.00  355.00  360.00  190.00
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.  
    0.      0.      0.      0.      0.   2420.00 2070.00 1880.00 1770.00 1660.00
 1550.00 1150.00 1300.00 1090.00  800.00  490.00  550.00  690.00  180.00  130.00
  140.00  930.00  840.00  210.00  220.00  360.00  740.00 1060.00 1110.00  750.00
  380.00  220.00  130.00  130.00  110.00  120.00  140.00  130.00  150.00  140.00
  130.00  125.00  140.00  340.00  330.00  330.00  250.00    0.      0.      0.  
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.  
    0.      0.   2370.00 1980.00 1820.00 1740.00 1500.00 1050.00  420.00  720.00
  814.00 1280.00  360.00  210.00  500.00  340.00  350.00  770.00 1060.00 1080.00
  430.00  350.00  510.00  920.00 1040.00  850.00  410.00  290.00  140.00  190.00
  120.00   14.00  120.00   98.00  135.00  140.00  145.00  120.00  120.00   95.00
  130.00  280.00  290.00  230.00    0.      0.      0.      0.      0.      0.  
    0.      0.      0.      0.      0.      0.      0.      0.      0.   2050.00
 1950.00 1900.00 1670.00 1100.00  600.00  520.00  550.00 1230.00 1250.00  950.00
 1000.00 1040.00  890.00  790.00  870.00  620.00  670.00 1220.00 1170.00 1110.00
  880.00  500.00  290.00  150.00  150.00  160.00  120.00   65.00    0.    110.00
  125.00  140.00  145.00  130.00  115.00  105.00  100.00  100.00  290.00  295.00
  290.00    0.      0.      0.      0.      0.      0.      0.      0.      0.  
    0.      0.      0.      0.      0.      0.   1890.00 1880.00 1760.00 1380.00
 1230.00 1040.00 1210.00 1250.00 1190.00 1210.00 1220.00 1190.00 1250.00 1340.00
 1290.00  880.00  750.00  890.00  880.00  330.00  450.00  340.00  148.00  109.00
  120.00   80.00   82.00   98.00  107.00   93.00  124.00  129.00  124.00  138.00
  122.00  120.00  109.00  117.00  157.00  265.00  274.00  245.00    0.      0.  
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.  
    0.      0.      0.   1620.00 1340.00 1360.00 1330.00 1040.00 1250.00 1340.00
 1320.00 1450.00 1640.00 1560.00 1050.00 1240.00 1460.00 1360.00 1100.00  980.00
 1040.00  840.00  250.00  130.00  130.00  117.00   96.00  135.00   73.00   74.00
```

```
   80.00     89.00    111.00    129.00    142.00    124.00    120.00    115.00    122.00    129.00
  131.00    159.00    240.00    263.00    193.00     0.        0.        0.        0.        0.
   0.        0.        0.        0.        0.        0.        0.        0.        0.        0.
  740.00   1060.00   1040.00    820.00    850.00   1460.00   1440.00   1560.00   1840.00   2016.00
 1750.00    540.00   1020.00   1620.00   1490.00   1320.00   1150.00    520.00    210.00    130.00
   95.00    100.00     84.00     80.00     93.00     49.00     23.00     74.00     87.00    117.00
  148.00    137.00    138.00    126.00    149.00    122.00    124.00    128.00    179.00    234.00
  303.00    208.00    184.00     0.        0.        0.        0.        0.        0.        0.
   0.        0.        0.       55.00     0.        0.        0.     1150.00   1130.00   1050.00
  680.00    490.00   1560.00   1550.00   1660.00   1840.00   1890.00   1880.00   1850.00   1790.00
 1730.00   1580.00   1240.00    870.00    140.00    160.00    120.00     95.00     88.00     67.00
   71.00     84.00     27.00     27.00     76.00     96.00    113.00    128.00    133.00    140.00
  137.00    128.00    117.00    111.00    122.00    117.00    181.00    277.00    235.00    202.00
   0.        0.        0.        0.        0.        0.        0.        0.      219.00    175.00
   88.00     0.        0.        0.     1150.00   1110.00   1030.00   1070.00   1250.00   1460.00
 1590.00   1630.00   1690.00   1820.00   1900.00   1860.00   1850.00   1120.00    350.00    220.00
  140.00     98.00     75.00     90.00    132.00     64.00     0.        0.        0.        0.
   64.00     76.00    107.00    106.00    124.00    129.00    148.00    149.00    131.00    115.00
   96.00     95.00     84.00    155.00    272.00    267.00    299.00    146.00     0.        0.
   0.        0.        0.        0.      347.00    612.00    466.00    137.00     45.00     0.
   0.     1110.00    620.00    580.00    420.00    260.00    840.00   1660.00   1670.00   1810.00
 1950.00   1980.00   1970.00   1650.00    360.00    190.00    130.00     70.00     80.00     0.
  100.00     96.00     0.        0.        0.        0.       43.00     47.00     69.00     95.00
  100.00    113.00    131.00    146.00    160.00    122.00     91.00     74.00     60.00     74.00
   89.00    124.00    181.00    210.00    294.00    270.00     0.        0.        0.        0.
  384.00    642.00    494.00    338.00    201.00     64.00     0.        0.     1080.00    630.00
  460.00    250.00    160.00    170.00   1280.00   1730.00   1810.00    980.00   2070.00   1970.00
 1190.00    160.00    150.00    110.00     85.00     25.00     60.00     73.00     93.00     0.
   0.        0.       20.00     40.00     60.00     60.00     60.00    100.00    111.00    107.00
  128.00    140.00     85.00     89.00     84.00     69.00     71.00     74.00     74.00     93.00
   85.00    168.00    296.00    358.00    321.00    420.00    433.00    329.00    133.00     58.00
   69.00    100.00     62.00      4.00     0.     1130.00    740.00    820.00    180.00    140.00
  180.00   1420.00   1780.00   1820.00    770.00   1950.00   2110.00   1640.00    160.00    150.00
  170.00     70.00     0.      126.00     0.        0.        0.        0.        0.        0.
   0.        0.        0.       42.00     87.00     84.00     78.00     87.00     89.00     87.00
   93.00     85.00     71.00     69.00     65.00     67.00     76.00     78.00    113.00    117.00
  117.00    202.00    210.00    128.00     51.00     20.00     18.00     0.       22.00     27.00
   40.00     0.     1050.00    650.00    270.00    170.00    180.00    260.00   1790.00   1980.00
 2010.00   2010.00   2270.00   2230.00   1980.00    450.00    130.00    130.00     70.00    129.00
   96.00     0.        0.        0.        0.        0.        0.        0.        0.        0.
   67.00     58.00     65.00     87.00     73.00     89.00     93.00     96.00     80.00     74.00
   64.00     65.00     64.00     60.00     53.00     53.00     54.00     60.00     53.00     47.00
   29.00     20.00     13.00     0.        0.        9.00     13.00     36.00     33.00    950.00
  410.00    210.00    190.00    250.00   1550.00   2150.00   2210.00   2250.00   2320.00   2320.00
 2120.00   1830.00   1410.00    125.00    135.00     50.00    170.00     80.00     65.00     0.
   0.        0.        0.        0.        0.        0.       40.00     67.00     60.00     69.00
   74.00     87.00     87.00     98.00     98.00     82.00     78.00     65.00     67.00     60.00
   62.00     51.00     51.00     51.00     40.00     32.00     31.00     0.        0.        0.
   0.        0.        9.00     11.00     26.00     31.00    440.00    280.00    200.00    280.00
 1590.00   2280.00   2320.00   2560.00   2560.00   2150.00   2050.00   1290.00   1650.00   1080.00
  140.00    150.00    160.00    100.00     43.00     34.00     0.        0.        0.        0.
   0.        0.        9.00     47.00     45.00     67.00     74.00     93.00     74.00    102.00
  102.00     98.00     80.00     73.00     74.00     54.00     69.00     72.00     54.00     49.00
   42.00     40.00     34.00     29.00     0.        0.        0.        0.        0.        0.
   16.00     29.00     33.00    320.00    250.00    270.00   1250.00   2320.00   2430.00   2550.00
 2610.00   2630.00   2620.00   2390.00   2080.00   1670.00    700.00    125.00    190.00    100.00
   70.00     60.00     43.00     71.00     0.        0.        0.        0.        0.       14.00
   51.00     53.00     56.00     76.00     71.00    102.00     73.00     85.00     98.00     74.00
  80.00   69.00   67.00   60.00   51.00   54.00   54.00   40.00   42.00   32.00
  29.00   0.      0.      0.      0.      0.      16.00   22.00   20.00   24.00
 350.00  420.00  750.00 2020.00 2550.00 2690.00 2750.00 2810.00 2750.00 2610.00
2440.00 2330.00 1940.00  240.00  120.00  160.00  118.00   67.00   53.00   0.
  56.00   0.      0.      0.      0.      0.      0.      0.      43.00   69.00
  78.00   98.00  107.00   95.00   98.00   80.00   82.00   62.00   54.00   49.00
  38.00   32.00   54.00   47.00   42.00   40.00   29.00   21.00   0.      0.
  0.      0.      9.00   16.00   12.00   0.      0.    1170.00 1510.00 2150.00
2490.00 2680.00 2770.00 2830.00 2850.00 2970.00 2710.00 2480.00 2250.00  860.00
 160.00   95.00   84.00   65.00   53.00   67.00  107.00   38.00   38.00   34.00
  0.      0.      0.      0.      0.      0.      64.00   95.00   67.00   65.00
  82.00   65.00   60.00   40.00   42.00   34.00   29.00   31.00   38.00   40.00
  54.00   49.00   32.00   29.00   18.00    7.00   0.      0.      0.      0.
  16.00   0.      0.      0.    2050.00 2320.00 2370.00 2670.00 2760.00 2850.00
2870.00 2870.00 2870.00 2710.00 2400.00 1580.00  160.00   73.00   76.00   42.00
  0.      0.      0.      0.     122.00   53.00   0.      0.      0.      0.
  0.      0.      0.      60.00   73.00   76.00   78.00   69.00   51.00   32.00
  32.00   31.00   27.00   34.00   38.00   45.00   40.00   45.00   49.00   38.00
  27.00   21.00   10.00   0.      0.      0.      0.      0.      0.      0.
  0.    2310.00 2380.00 2660.00 2740.00 2830.00 2890.00 2910.00 2870.00 2830.00
2810.00 2030.00  250.00  106.00   84.00   51.00   0.      0.      0.      0.
  0.      60.00  129.00   0.      42.00   27.00   0.      0.      0.      0.
  38.00   60.00   67.00   73.00   74.00   40.00   25.00   25.00   23.00   23.00
  42.00   56.00   47.00   38.00   38.00   38.00   42.00   31.00   20.00    7.00
  0.      0.      0.      0.      0.      0.      0.      0.    2450.00 2550.00
2730.00 2770.00 2810.00 2940.00 2940.00 2950.00 2750.00  950.00  280.00  120.00
  91.00   74.00   47.00   0.      0.      0.      0.      0.      0.      95.00
  84.00   45.00   32.00   0.      0.      0.      0.      0.      0.      56.00
  65.00   47.00   32.00   18.00   18.00   25.00   40.00   38.00   53.00   53.00
  38.00   42.00   40.00   38.00   36.00   21.00   10.00    5.00   0.      0.
  0.      0.      0.      0.      0.    2790.00 2780.00 2840.00 2950.00 2850.00
2920.00 2120.00 1880.00  950.00  310.00  180.00   51.00   0.      0.      0.
  0.      0.      0.      0.      0.      25.00   80.00   65.00   40.00   32.00
  23.00   0.      0.      0.      0.      0.      38.00   49.00   40.00   47.00
  58.00   40.00   42.00   38.00   31.00   49.00   45.00   42.00   38.00   36.00
  32.00   38.00   34.00   16.00    5.00   0.      0.      0.      0.      0.
  0.      0.    2963.00 2999.00 2968.00 1719.00 1061.00  293.00  384.00  329.00
 293.00  192.00  146.00   93.00   0.      0.      0.      0.      0.      0.
  0.      0.      69.00   74.00   53.00   40.00   32.00   23.00   0.      0.
  0.      0.      0.      0.      25.00   42.00   32.00   40.00   32.00   49.00
  56.00   38.00   42.00   40.00   34.00   31.00   27.00   21.00   21.00   20.00
   9.00   0.      0.      0.      0.      0.      0.      0.      0.    3182.00
2889.00 1317.00  457.00  174.00  201.00  274.00  311.00  219.00  155.00  113.00
  56.00   0.      0.      0.      0.      0.      0.      0.      0.      73.00
  80.00   54.00   38.00   27.00   21.00   0.      0.      0.      0.      0.
  0.      14.00   32.00   21.00   29.00   34.00   31.00   32.00   31.00   29.00
  27.00   20.00   21.00   16.00    5.00   0.      0.      0.      0.      0.
  0.      0.      0.      0.      0.      0.    3164.00 2414.00  658.00  219.00
 185.00  238.00  347.00  320.00  182.00  137.00  128.00  107.00   45.00   0.
  0.      0.      0.      0.      0.      0.      69.00   84.00   56.00   0.
  0.      0.      0.      0.      0.      0.      0.      0.       9.00    9.00
  21.00   32.00   29.00   32.00   31.00   31.00   27.00   18.00    5.00   0.
  0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
  0.      0.      0.    3237.00 1733.00  640.00  256.00  219.00  296.00  457.00
 311.00  174.00  128.00  120.00  100.00   64.00   0.      0.      0.      0.
  0.      0.      0.      64.00   76.00   56.00   10.00   0.      0.      0.
  0.      0.      0.      0.      0.      0.      0.      0.      29.00   38.00
  40.00   32.00   25.00   25.00   0.      0.      0.      0.      0.      0.
```

```
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
  3145.00  1865.00   561.00   329.00   311.00   402.00   567.00   475.00   192.00   135.00
   119.00    56.00    32.00     0.       0.       0.       0.       0.       0.      42.00
    69.00    74.00    51.00    18.00     0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.      25.00    42.00    40.00    32.00    29.00
    20.00     0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.    3237.00  1591.00   658.00
   378.00   384.00   494.00   768.00   622.00   347.00   284.00   121.00    43.00     0.  
     0.       0.       0.       0.       0.      43.00    62.00    78.00    67.00    31.00
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.      27.00    43.00    42.00    29.00    23.00    18.00     0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.    3145.00  2268.00   494.00   357.00   457.00   805.00
  1287.00   933.00   603.00   201.00   143.00    58.00     0.       0.       0.       0.  
    58.00    60.00    65.00    84.00    82.00    49.00     0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.      20.00    38.00
    40.00    32.00    25.00    16.00     0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.    4444.00  3460.00  2524.00   512.00   457.00   567.00  1024.00  1554.00  1261.00
   878.00   201.00   155.00   133.00    60.00    67.00    87.00    85.00    78.00    82.00
   104.00    85.00    60.00    42.00    25.00    23.00    16.00     0.       0.       0.  
     0.       0.       0.       0.       9.00    20.00    31.00    31.00    25.00    14.00
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.    4224.00  3200.00
   859.00  1000.00  1408.00  1957.00  1572.00  1737.00  1188.00   274.00   168.00   137.00
    97.00    98.00   106.00   100.00    98.00    95.00    96.00   106.00    85.00    73.00
    53.00    25.00     0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.      18.00    34.00    21.00    10.00     0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.    4407.00  3932.00  1975.00   859.00  1170.00
  2213.00  2103.00  2060.00   878.00   823.00   170.00   149.00   124.00   122.00   102.00
   102.00    84.00   107.00    96.00    96.00    84.00    69.00    45.00     0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.      38.00    36.00
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.    4316.00  4115.00  3548.00  2656.00  2687.00  2486.00  2431.00  2266.00
  2102.00  1041.00   179.00   146.00   111.00    96.00   105.00    91.00   107.00   111.00
   100.00    91.00    97.00    53.00     0.       0.       0.      21.00    34.00    36.00
    29.00    31.00    40.00    49.00    47.00    42.00    29.00     0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.    4533.00
  4295.00   410.00  3820.00  3436.00  2705.00  2412.00  2303.00  1901.00   768.00   192.00
   142.00   137.00   102.00    96.00   133.00    85.00   120.00   102.00    82.00    34.00
     0.      42.00    58.00    43.00    62.00    62.00    58.00    60.00    64.00    51.00
    51.00    42.00    36.00    18.00     0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.    4679.00  4443.00  4259.00  4131.00
  3601.00  2468.00  2449.00  1974.00  1152.00   405.00   174.00   128.00   137.00   149.00
   124.00   142.00   132.00   119.00   108.00    95.00    78.00    85.00    74.00    76.00
    74.00    67.00   102.00    65.00    62.00    43.00    49.00    38.00    27.00     0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.    4773.00  4663.00  4407.00  4243.00  3932.00  2021.00  1536.00
  1298.00   896.00   329.00   154.00   128.00   137.00   144.00   138.00   137.00   130.00
   124.00   110.00   100.00   100.00    96.00    93.00    85.00   100.00    74.00    60.00
    31.00     0.      12.00    23.00    18.00     0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
     0.       0.       0.       0.       0.       0.       0.       0.       0.       0.  
```

```
 4828.00 4082.00 4626.00 4999.00 4426.00 3584.00 1742.00  695.00  805.00  384.00
  155.00  146.00  157.00  153.00  144.00  138.00  140.00  130.00  128.00  122.00
  113.00  107.00  102.00   91.00   87.00   74.00   56.00   38.00   12.00    0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.   4810.00 4810.00 4773.00
 4608.00 4554.00 3932.00 2231.00 1627.00 2249.00 1554.00  512.00  311.00  201.00
  174.00  155.00  143.00  143.00  146.00  139.00  121.00  115.00  109.00  102.00
   87.00   71.00   34.00   29.00   29.00   10.00    0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.   4682.00 4896.00 4517.00 4535.00 4462.00 4115.00
 3072.00 2020.00 2080.00 2231.00 2322.00 1298.00  439.00  165.00  175.00  170.00
  159.00  161.00  148.00  135.00  122.00  151.00   76.00    0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.   4846.00 4887.00 4896.00 4700.00 4407.00 4499.00 3219.00 3566.00 2871.00
 3255.00 2875.00 2798.00 1330.00  950.00  420.00  475.00  162.00  151.00  142.00
  131.00  120.00  110.00   42.00    0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.   4840.00 4850.00
 4660.00 4600.00 4650.00 4450.00 4150.00 4000.00 3650.00 4410.00 3850.00 3850.00
 3820.00 3650.00 2720.00 1950.00  550.00  179.00  148.00  144.00  130.00  115.00
   56.00   16.00    0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.   4780.00 4830.00 4750.00 4820.00 4750.00
 4520.00 4450.00 4620.00 4670.00 4580.00 4530.00 4230.00 4380.00 4260.00 2470.00
 2150.00 2350.00  676.00  460.00  148.00  135.00  130.00  115.00   95.00   74.00
   31.00    7.00    0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.   4780.00 4840.00 4750.00 4820.00 4680.00 4750.00 4630.00 4680.00
 4740.00 4670.00 4620.00 4550.00 4530.00 4450.00 4340.00 3750.00 4180.00 4330.00
 4120.00 2460.00  713.00  135.00  129.00  117.00  107.00   77.00   35.00    7.00
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.   4670.00
 4790.00 4820.00 4780.00 4730.00 4480.00 4360.00 4720.00 4750.00 4760.00 4780.00
 4640.00 4620.00 4180.00 3940.00 4310.00 4580.00 4630.00 4610.00 4130.00 2960.00
  804.00  153.00  129.00  124.00   89.00   49.00    7.00    0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.   4510.00 4480.00 4590.00 4810.00
 4060.00 4440.00 4150.00 4580.00 4850.00 4780.00 4750.00 4720.00 4740.00 4730.00
 4780.00 4790.00 4720.00 4650.00 4610.00 4610.00 4220.00 2380.00  512.00  159.00
  135.00  115.00   93.00   42.00   12.00    0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.   4120.00 4540.00 4660.00 4820.00 4820.00 4730.00 4250.00
 4460.00 4870.00 4780.00 4800.00 4800.00 4810.00 4750.00 4920.00 4920.00 4770.00
 4770.00 4750.00 4730.00 4560.00 4540.00 2560.00  658.00  140.00  128.00  111.00
   49.00   29.00    7.00    0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
    0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
 4330.00 4630.00 4620.00 4810.00 4740.00 4820.00 4860.00 4870.00 4820.00 4820.00
```

```
4810.00 4810.00 4920.00 4840.00 4860.00 4860.00 4820.00 4810.00 4810.00 4720.00
4690.00 4670.00 4230.00 3150.00  155.00  126.00  110.00   86.00   46.00    9.00
     0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
     0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
     0.      0.      0.      0.      0.      0.      0. 4580.00 4840.00 4670.00
4160.00 4840.00 4660.00 4830.00 4820.00 4570.00 4580.00 3770.00 3650.00 4430.00
4880.00 4910.00 4870.00 4840.00 4830.00 4830.00 4770.00 4630.00 4550.00 4420.00
4100.00 2450.00  201.00  118.00  106.00   67.00   20.00      0.      0.      0.
     0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
     0.      0.      0.      0.      0.      0.      0.      0.      0.      0.
     0.      0.      0.      0.
```

```
      PROGRAM SETUPT(UNIT5=INPUT,UNIT6=OUTPUT,UNIT23=CSPBIT)
C*********************************************************************
C*                                                                   *
C*    PROGRAM SETUPT                                                 *
C*    SETS UP TIDAL INPUT DATA FOR SEA MODEL IN A FILE               *
C*    MODIFIED 12/06/83 FOR 6 CONSTITUENTS                           *
C*                                                                   *
C*********************************************************************
      DIMENSION Z1(6,150),Z2(6,150),U1(6,100),U2(6,100),
     &          V1(6,100),V2(6,100)
      DIMENSION ATC(6,4),BTC(6,3),ICTC(6,4),SIG(6)
C
C     WRITE TIDAL BOUNDARY INPUT DATA TO DISC
C
      NR=5
      NW=6
      NBOE=23
      READ(NR,102) IOBZ,IOBU,IOBV
      READ(NR,102) NCON
  102 FORMAT(20I4)
      WRITE(NBOE) NCON
C
C     READ BOUNDARY DATA FROM CARDS
C
      DO 1 ICON=1,NCON
      READ(NR,100) SIG(ICON)
      READ(NR,100) (ATC(ICON,J),J=1,4)
      READ(NR,100) (BTC(ICON,J),J=1,3)
      READ(NR,102) (ICTC(ICON,J),J=1,4)
      READ(NR,100) (Z1(ICON,J),J=1,IOBZ)
      READ(NR,100) (Z2(ICON,J),J=1,IOBZ)
      READ(NR,100) (U1(ICON,J),J=1,IOBU)
      READ(NR,100) (U2(ICON,J),J=1,IOBU)
      READ(NR,100) (V1(ICON,J),J=1,IOBV)
      READ(NR,100) (V2(ICON,J),J=1,IOBV)
    1 CONTINUE
  100 FORMAT(8F10.6)
C
C     WRITE BOUNDARY DATA TO DISC AND PRINT OUT
C
      WRITE(NW,101)
  101 FORMAT(1H1,45H  TIDAL INPUT BOUNDARY DATA WRITTEN TO DISC  //)
      DO 2 ICON=1,NCON
      WRITE(NBOE) SIG(ICON)
      WRITE(NBOE) (ATC(ICON,J),J=1,4)
      WRITE(NBOE) (BTC(ICON,J),J=1,3)
      WRITE(NBOE) (ICTC(ICON,J),J=1,4)
      WRITE(NBOE) (Z1(ICON,J),J=1,IOBZ)
      WRITE(NBOE) (Z2(ICON,J),J=1,IOBZ)
      WRITE(NBOE) (U1(ICON,J),J=1,IOBU)
      WRITE(NBOE) (U2(ICON,J),J=1,IOBU)
      WRITE(NBOE) (V1(ICON,J),J=1,IOBV)
      WRITE(NBOE) (V2(ICON,J),J=1,IOBV)
      WRITE(NW,103) ICON
  103 FORMAT(1H0,17H  CONSTANT NUMBER,I4///)
      WRITE(NW,100) SIG(ICON)
      WRITE(NW,100) (ATC(ICON,J),J=1,4)
      WRITE(NW,100) (BTC(ICON,J),J=1,3)
      WRITE(NW,102) (ICTC(ICON,J),J=1,4)
      WRITE(NW,100) (Z1(ICON,J),J=1,IOBZ)
      WRITE(NW,100) (Z2(ICON,J),J=1,IOBZ)
      WRITE(NW,100) (U1(ICON,J),J=1,IOBU)
      WRITE(NW,100) (U2(ICON,J),J=1,IOBU)
      WRITE(NW,100) (V1(ICON,J),J=1,IOBV)
      WRITE(NW,100) (V2(ICON,J),J=1,IOBV)
    2 CONTINUE
      STOP
      END
```

```
103  50  68                                    Input data for program SETUPT
  2
28.984104
 1.004000 -0.037300  0.000200
-0.037400
 -2   2
-0.507304 -0.487000 -0.468855 -0.450572 -0.434569 -0.421702 -0.413446 -0.411788
-0.438135 -0.037971  0.045220 -0.002670 -0.003491  0.012246  0.022660  0.034732
 0.053830  0.074908  0.097299  0.120785  0.145347  0.163507  0.181485  0.199525
 0.216499  0.227131  0.237933  0.248355  0.252712  0.257613  0.263060  0.270140
 0.278310  0.293575 -0.757306 -0.787736 -0.825640 -0.844882 -0.871121 -0.544917
-0.690545 -0.579912 -0.665080 -0.611627 -0.645775 -0.643117 -0.615068 -0.672078
-0.600879 -0.701572 -0.563554 -0.730555 -0.538697 -0.755885 -0.513949 -0.781000
-0.504747 -0.520233 -0.535146 -0.535706 -0.550245 -0.802878 -0.824497 -0.841944
-0.858272 -0.872426 -0.883372 -0.894262 -0.900318 -0.904106 -0.905620 -0.912063
-0.911501 -0.910494 -0.909891 -0.907781 -0.564784 -0.560977 -0.535638 -0.504201
-0.455905 -0.251571 -0.253650 -0.257809 -0.418938 -0.220531 -0.385843 -0.198670
-0.341845 -0.351612 -0.178943 -0.321608 -0.327455 -0.336226 -0.344997 -0.162641
-0.310582 -0.147119 -0.144718 -0.139045 -0.131416 -0.116611 -0.097854
 0.026587  0.000001 -0.032785 -0.063323 -0.100327 -0.145203 -0.201650 -0.267417
-0.342307 -0.195344 -0.096975 -0.152977 -0.199970 -0.233679 -0.259011 -0.282876
-0.305291 -0.324465 -0.339326 -0.350788 -0.359748 -0.367246 -0.372101 -0.375253
-0.374989 -0.378011 -0.380774 -0.382434 -0.389144 -0.396692 -0.405079 -0.415981
-0.428561 -0.435245  0.473219  0.473321  0.457662  0.468327  0.482872  0.009512
 0.483526 -0.010121  0.519619 -0.021357  0.541871 -0.033703  0.573561 -0.035221
 0.622230 -0.024498  0.648296 -0.025510  0.665238 -0.013193  0.682035  0.000001
 0.720854  0.742971  0.764269  0.794219  0.815773  0.014015  0.028793  0.058875
 0.090210  0.122613  0.155764  0.173828  0.207856  0.242256  0.276878  0.296349
 0.331761  0.367865  0.405111  0.442755  0.837328  0.863831  0.891455  0.909605
 0.934746  1.183558  1.193339  1.212903  0.940952  1.250705  0.955000  1.254363
 0.992794  1.021160  1.257329  1.051934  1.071061  1.099750  1.128439  1.269625
 1.159111  1.271517  1.332162  1.393077  1.444032  1.515519  1.647095
 0.003190 -0.025802 -0.005260 -0.024799 -0.007970 -0.016689 -0.006094 -0.008175
-0.001675 -0.001670  0.009273  0.000352  0.000829  0.011636 -0.006007  0.016965
-0.009712  0.018449 -0.012246  0.029800 -0.012523 -0.010152 -0.008447 -0.007277
-0.007743 -0.012611 -0.016178 -0.016618 -0.017612 -0.017538 -0.017800 -0.017873
-0.015194 -0.011051 -0.001817  0.017463  0.021295  0.019895  0.010532 -0.004637
 0.006856  0.016729  0.017037  0.027575  0.020448  0.029434  0.019230  0.032819
-0.009637  0.027389
-0.091344  0.031863 -0.075216  0.028528 -0.064912  0.016116 -0.057981  0.007360
-0.047971  0.004851 -0.043625  0.004105 -0.047493 -0.001223 -0.037927  0.004293
-0.025300  0.001374 -0.018156 -0.000000 -0.015465 -0.014499 -0.014058 -0.013686
-0.011059 -0.005353  0.000848  0.006379  0.011437  0.015792  0.019088  0.022071
 0.025286  0.027352  0.016198  0.034872  0.032792  0.025191  0.011098  0.023854
 0.000781  0.040588 -0.003004  0.019023 -0.006644  0.018752  0.014023  0.016867
 0.054657  0.021630
-0.049032 -0.059351 -0.072982 -0.087687 -0.115794 -0.142244 -0.178687 -0.243344
-0.310569 -0.365661 -0.461861 -0.281139 -0.151649 -0.064433 -0.022922 -0.001798
 0.011217  0.014159  0.016841  0.016736  0.014909  0.009794  0.003660  0.000677
 0.000991  0.000232 -0.001718 -0.000286  0.001839  0.003744  0.007256  0.012586
 0.015399  0.016863 -0.028887 -0.031713 -0.037326 -0.052160 -0.068533 -0.034013
-0.034549 -0.021881 -0.011094 -0.008751 -0.008638 -0.007403 -0.002502  0.017245
 0.032063  0.064222  0.041684 -0.013906  0.028369 -0.008153  0.006369  0.018572
 0.001134  0.010973  0.028292  0.070305  0.015924  0.057033  0.003680  0.143320
 0.144149  0.123017  0.139761  0.211050
-0.081602 -0.094981 -0.112382 -0.135026 -0.159376 -0.195782 -0.245941 -0.334932
-0.516871 -0.237462 -0.115154 -0.235903 -0.208726 -0.159476 -0.129995 -0.102984
-0.079816 -0.066612 -0.055083 -0.043598 -0.036902 -0.032036 -0.026044 -0.019388
```

```
-0.014165 -0.013298 -0.013995 -0.016398 -0.021020 -0.026638 -0.034137 -0.034581
-0.028961 -0.024083  0.038335  0.034008  0.021550  0.013976  0.004792  0.048576
 0.028990  0.024301  0.034143  0.037903  0.037416  0.038087  0.034910  0.022312
 0.024249  0.035746  0.008784  0.042798 -0.005514  0.030427  0.037260 -0.011605
 0.032480  0.035231  0.044581  0.085896 -0.013843  0.056242 -0.029975 -0.089557
-0.125308 -0.106938 -0.105318 -0.041024
30.0
 1.000000  0.0       0.0
 0.0
  0   0
-0.180492 -0.168530 -0.156510 -0.142538 -0.130270 -0.115753 -0.102295 -0.082414
-0.074783  0.012674  0.017893  0.017714  0.029237  0.041581  0.050291  0.061571
 0.071539  0.083465  0.093768  0.103046  0.112430  0.121132  0.129783  0.135768
 0.140894  0.145928  0.149256  0.152576  0.154290  0.158482  0.160214  0.163679
 0.166290  0.171473 -0.342165 -0.352653 -0.363001 -0.371951 -0.383886 -0.187292
-0.319000 -0.194575 -0.317564 -0.203768 -0.313274 -0.212961 -0.309146 -0.221387
-0.313007 -0.233144 -0.305233 -0.240916 -0.299323 -0.251376 -0.294056 -0.261154
-0.296699 -0.305971 -0.315242 -0.321256 -0.329541 -0.271021 -0.283531 -0.291858
-0.303437 -0.317861 -0.328060 -0.334428 -0.342428 -0.350093 -0.357406 -0.364352
-0.370916 -0.377081 -0.383816 -0.390137 -0.337825 -0.342579 -0.337048 -0.331988
-0.322295 -0.312541 -0.309817 -0.318905 -0.309231 -0.317811 -0.300145 -0.313153
-0.294538 -0.301532 -0.307691 -0.298865 -0.306412 -0.312449 -0.326335 -0.302752
-0.319601 -0.295780 -0.312142 -0.321990 -0.336151 -0.346645 -0.378541
-0.136010 -0.141413 -0.145947 -0.152853 -0.160869 -0.171611 -0.184545 -0.203980
-0.230155 -0.103225 -0.058526 -0.083338 -0.095630 -0.102917 -0.107851 -0.111077
-0.114487 -0.114881 -0.115795 -0.114444 -0.112430 -0.109069 -0.105097 -0.102309
-0.098655 -0.094768 -0.093266 -0.091677 -0.092707 -0.091500 -0.092500 -0.094500
-0.099918 -0.099000 -0.023926 -0.030853 -0.038152 -0.039093 -0.040348 -0.151666
 0.000001 -0.163268  0.016643 -0.170981  0.032927 -0.178695  0.048964 -0.185765
 0.066532 -0.188796  0.081787 -0.195089  0.091513 -0.196396  0.101252 -0.196793
 0.119874  0.123621  0.127367  0.136366  0.139882 -0.196908 -0.191243 -0.189534
-0.182323 -0.169009 -0.160005 -0.155946 -0.145351 -0.134387 -0.123064 -0.111393
-0.099386 -0.087055 -0.074606 -0.061791  0.143399  0.152527  0.164389  0.176522
 0.193655  0.312541  0.320826  0.318905  0.208579  0.340811  0.218069  0.347792
 0.238513  0.244177  0.353959  0.259800  0.266360  0.271609  0.273828  0.360807
 0.298034  0.365259  0.385465  0.412130  0.430254  0.460015  0.502342
 0.018280 -0.015557  0.012557 -0.014471  0.009171 -0.009616  0.008580 -0.004888
 0.008035 -0.001693  0.009970 -0.000699  0.008220  0.004293  0.004308  0.004806
 0.001032  0.005690 -0.001084  0.009644 -0.001641 -0.001112 -0.000729 -0.000515
-0.001169 -0.003914 -0.006336 -0.007419 -0.008615 -0.009382 -0.010083 -0.010678
-0.010392 -0.009559 -0.004507 -0.002927 -0.001705 -0.000945 -0.000964 -0.006367
 0.000805 -0.004355  0.005458  0.004323  0.007179  0.005348  0.002450  0.007057
-0.015494  0.004352
-0.025160  0.006603 -0.022653  0.005267 -0.019667  0.002397 -0.016840  0.001310
-0.013372  0.001421 -0.009970  0.001036 -0.012187  0.001007 -0.010663  0.003948
-0.008407  0.003285 -0.006845  0.004153 -0.006124 -0.005233 -0.005189 -0.004903
-0.004076 -0.003285 -0.001698 -0.000129  0.001211  0.002690  0.003870  0.004532
 0.006000  0.007741  0.005768  0.011741  0.011979  0.008990  0.002511  0.004295
-0.001108  0.015188  0.000594  0.011942 -0.000125  0.012068  0.006450  0.012322
 0.011675  0.012568
 0.002850  0.003373  0.003299  0.002941 -0.000000 -0.000000 -0.000000  0.002548
 0.018389 -0.052680 -0.101561 -0.027894  0.004694  0.017279  0.022398  0.023285
 0.021648  0.019457  0.017694  0.015329  0.013068  0.010115  0.006581  0.003797
 0.002444  0.001303  0.000115  0.000547  0.001719  0.002770  0.004306  0.005633
 0.006201  0.006585 -0.017723 -0.017309 -0.016190 -0.019759 -0.024209 -0.021384
-0.018194 -0.012764 -0.010943 -0.010628 -0.010298 -0.010171 -0.008087 -0.001035
 0.002718  0.008729  0.009218 -0.014478  0.009339 -0.009587 -0.006948  0.008158
-0.007686 -0.005898 -0.003421 -0.002857  0.008175  0.000697  0.007561  0.065303
```

```
 0.073595  0.062854  0.068190  0.078954
-0.032576 -0.038553 -0.047185 -0.056123 -0.068800 -0.084800 -0.106000 -0.145978
-0.210197 -0.137236 -0.121035 -0.120822 -0.089577 -0.056518 -0.040407 -0.028754
-0.020187 -0.016327 -0.012389 -0.008850 -0.006948 -0.006077 -0.005721 -0.003797
-0.002050 -0.001609 -0.002197 -0.003102 -0.004050 -0.005680 -0.007166 -0.007210
-0.005988 -0.004785  0.007891  0.005624  0.000565 -0.004200 -0.010276  0.010896
 0.003867  0.004395  0.009513  0.011006  0.011043  0.011296  0.011550  0.010951
 0.014955  0.026394  0.013564  0.012149  0.005813  0.008632  0.013068  0.001295
 0.010977  0.013246  0.019807  0.041903 -0.000286  0.030692 -0.008398  0.003560
-0.007735 -0.006606 -0.001190  0.033840
```

```
      PROGRAM METSET(UNIT5=INPUT,UNIT6=OUTPUT,UNIT10[,,4]=WINDFMF,
     1UNIT11[,,4]=FMFCSP,UNIT12[,,4]=CSPID1)
C*******************************************************************
C*                                                                 *
C*    MET DATA SETUP PROGRAM - METSET     CYBER  13/08/82          *
C*    COMPUTES ALL PARAMETERS FOR                                  *
C*    LINEAR INTERPOLATION FROM MET POINTS TO SEA POINTS           *
C*    MODIFIED 6/10/82 FOR EXTENDED AND PATCHED VERSIONS           *
C*                                                                 *
C*******************************************************************
      DIMENSION CHIU(100),CHIV(100),PHIU(100),
     &          PHIV(100),IMP(100),JMP(100),CHIM(100),PHIM(100),
     &          AU(100),BU(100),AV(100),BV(100),ID(8704)
      DIMENSION IDD(10),DAT(7)
      DIMENSION BITW(8704),BITP(8704)
      DIMENSION IIMU(3072),IIMV(3072),IIMZ(3072),
     &          AUX(3072),AVX(3072),AZX(3072),BUX(3072),BVX(3072),
     &          BZX(3072),IDM(512)
      DIMENSION IIMUX(3072),IIMVX(3072),IIMZX(3072),
     &      AUXX(3072),AVXX(3072),AZXX(3072),BUXX(3072),BVXX(3072),
     &      BZXX(3072),LABX(3072),IXST(10),JXST(10),NCT(10),NRT(10),
     &      IF(10),JF(10)
      COMMON/HEAD/A(512)
      EQUIVALENCE (A(1),IDD(1)),(A(11),DAT(1))
      COMMON/BUMIT/BITIM(32768)
      EQUIVALENCE (BITW(1),BITIM(1)),(BITP(1),BITIM(8705))
      COMMON/BUMRI/DAM(28160)
      EQUIVALENCE (IIMU(1),DAM(1)),(IIMV(1),DAM(3073)),
     &            (IIMZ(1),DAM(6145)),(AUX(1),DAM(9217)),
     &            (AVX(1),DAM(12289)),(AZX(1),DAM(15361)),
     &            (BUX(1),DAM(18433)),(BVX(1),DAM(21505)),
     &            (BZX(1),DAM(24577)),(IDM(1),DAM(27649))
      BIT BITW,BITWD,BITIM
      BIT BITP,BITPD
      DESCRIPTOR BITWD
      DESCRIPTOR ISEAD,IIMUD,IIMVD
      DESCRIPTOR AUXD,BUXD,AVXD,BVXD
      DESCRIPTOR BITPD
      NR=5
      NW=6
      NRM=10
      NRP=11
      NWM=12
C
C     READ DATA FOR SEA MODEL RECTANGLE FROM CARDS
C
      READ(NR,100) NRX,NCX
      WRITE(NW,100) NRX,NCX
  100 FORMAT(16I5)
      READ(NR,114) CHIN,PHIN
      WRITE(NW,114) CHIN,PHIN
  114 FORMAT(8F10.5)
      READ(NR,113) DX,DY
      WRITE(NW,113) DX,DY
  113 FORMAT(6F12.9)
      ISX=NRX*NCX
C
C     CONVERT DX,DY TO DEGREES
```

```
C
      DX=DX*180./3.1415926535
      DY=DY*180./3.1415926535
C
      DO 30 I=1,NCX
      CHIV(I)=CHIN+(I-1)*DX
   30 CHIU(I)=CHIV(I)+0.5*DX
C
      DO 31 J=1,NRX
      PHIU(J)=PHIN-(J-1)*DY
   31 PHIV(J)=PHIU(J)-0.5*DY
C
C     CHIU,PHIU CONTAIN LONG,LAT OF SEA MODEL U POINTS
C     CHIV,PHIV CONTAIN LONG,LAT OF SEA MODEL V POINTS
C
C
C     READ IN AND PRINT HEADER FROM WIND DATA
C
      CALL Q7BUFIN(NRM,A,1,'SMALL')
      CALL Q7WAIT(NRM,A,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,205) ISTAT
  205 FORMAT(10X,' ABNORMAL END OF WHEAD STAT=',I5)
      WRITE(NW,100) (IDD(I),I=1,10)
      WRITE(NW,102) (DAT(I),I=1,7)
  102 FORMAT(8F10.6)
      NSPAG=(2*IDD(3)+1023)/1024
      NWEL=IDD(3)
      NCXM=IDD(1)
      NRXM=IDD(2)
      DXM=DAT(3)
      DYM=DAT(2)
      DXMI=1./DXM
      DYMI=1./DYM
      PHINM=DAT(4)
      CHINM=DAT(5)-360.
C
C     FIND LIMITS OF DATA REQUIRED FOR INTERPOLATION  -  WINDS
C
      IS=IFIX((CHIN-CHINM)*DXMI)-1
      JS=IFIX((PHINM-PHIN)*DYMI)-1
      IE=IFIX((CHIN+(NCX-1)*DX+0.5*DX-CHINM)*DXMI)+4
      JE=IFIX((PHINM-(PHIN-(NRX-1)*DY-0.5*DY))*DYMI)+4
      WRITE(NW,100) IS,IE,JS,JE
      NCXMS=IE-IS+1
      NRXMS=JE-JS+1
C
C     SET UP MASK FOR WIND DATA REQUIRED
C
      ASSIGN BITWD,BITW(1;NWEL)
      ASSIGN ISEAD,ID(1;NWEL)
      BITWD=BITWD.AND..NOT.BITWD
      ISEAD=0
      DO 1 JR=JS,JE
      DO 1 IC=IS,IE
      II=(JR-1)*NCXM+IC
      ID(II)=1
    1 CONTINUE
      BITWD=ISEAD.EQ.1
      IWO=Q8SCNT(BITWD)
C
C     COMPUTE ALL PARAMETERS FOR LINEAR INTERPOLATION OF EAST WIND
C     STRESS TO SEA MODEL U-POINTS
C
C     EVALUATE FOR U POINTS
C
      DO 32 I=1,NCX
      IMP(I)=1+IFIX((CHIU(I)-CHINM)*DXMI)
   32 CHIM(I)=CHINM+(IMP(I)-1)*DXM
C
      DO 33 J=1,NRX
      JMP(J)=1+IFIX((PHINM-PHIU(J))*DYMI)
   33 PHIM(J)=PHINM-(JMP(J)-1)*DYM
C
C     CHIM,PHIM CONTAIN LONG,LAT OF U STRESS POINTS IN SEA MODEL SPACE
C
      AU(1;NCX)=(CHIU(1;NCX)-CHIM(1;NCX))*DXMI
      BU(1;NRX)=(PHIM(1;NRX)-PHIU(1;NRX))*DYMI
C
C     AU,BU INTERPOLATION FACTORS FOR EACH SEA MODEL U POINT
C
      K=0
      DO 34 J=1,NRX
      B=BU(J)
      DO 34 I=1,NCX
      K=K+1
      AUXX(K)=AU(I)
      BUXX(K)=B
   34 IIMUX(K)=(JMP(J)-JS)*NCXMS+IMP(I)-IS+1
      WRITE(NW,96)
      CALL ARPRIN(DUM,IIMUX,NCX,NRX,1)
C
C     IIMUX IS 1-D ARRAY OF MET POINT NUMBERS FOR EACH SEA MOD U-POINT
C     LESS 1 TO GIVE CORRECT ADDRESS RELATIVE TO START LOCN IN Q8VXTOV
C
C     COMPUTE ALL PARAMETERS FOR LINEAR INTERPOLATION OF NORTH WIND
C     STRESS TO SEA MODEL V-POINTS
C
      DO 36 I=1,NCX
      IMP(I)=1+IFIX((CHIV(I)-CHINM)*DXMI)
   36 CHIM(I)=CHINM+(IMP(I)-1)*DXM
C
      DO 37 J=1,NRX
      JMP(J)=1+IFIX((PHINM-PHIV(J))*DYMI)
   37 PHIM(J)=PHINM-(JMP(J)-1)*DYM
C
C     CHIM,PHIM CONTAIN LONG,LAT OF V STRESS POINTS IN SEA MODEL SPACE
C
      AV(1;NCX)=(CHIV(1;NCX)-CHIM(1;NCX))*DXMI
      BV(1;NRX)=(PHIM(1;NRX)-PHIV(1;NRX))*DYMI
C
C     AV,BV INTERPOLATION FACTORS FOR EACH SEA MODEL V POINT
C
      K=0
      DO 38 J=1,NRX
      B=BV(J)
      DO 38 I=1,NCX
      K=K+1
      AVXX(K)=AV(I)
      BVXX(K)=B
   38 IIMVX(K)=(JMP(J)-JS)*NCXMS+IMP(I)-IS+1
      WRITE(NW,96)
      CALL ARPRIN(DUM,IIMVX,NCX,NRX,1)
C
C     IIMVX IS 1-D ARRAY OF MET POINT NUMBERS FOR EACH SEA MOD V-POINT
C     LESS 1 TO GIVE CORRECT ADDRESS RELATIVE TO START LOCN IN Q8VXTOV
C
C
C     READ IN AND PRINT HEADER FROM PRESSURE DATA
C
      CALL Q7BUFIN(NRP,A,1,'SMALL')
      CALL Q7WAIT(NRP,A,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,206) ISTAT
  206 FORMAT(10X,' ABNORMAL END OF PHEAD STAT=',I5)
      WRITE(NW,100) (IDD(I),I=1,10)
      WRITE(NW,102) (DAT(I),I=1,7)
      NPEL=IDD(3)
      NCXP=IDD(1)
      NRXP=IDD(2)
      NSPAP=(NPEL+1023)/1024
      PHINM=DAT(4)
      CHINM=DAT(5)-360.
C
C     FIND LIMITS OF DATA REQUIRED FOR INTERPOLATION  -  PRESSURES
C
      IS=IFIX((CHIN-CHINM)*DXMI)-1
      JS=IFIX((PHINM-PHIN)*DYMI)-1
      IE=IFIX((CHIN+(NCX-1)*DX+0.5*DX-CHINM)*DXMI)+4
      JE=IFIX((PHINM-(PHIN-(NRX-1)*DY-0.5*DY))*DYMI)+4
      WRITE(NW,100) IS,IE,JS,JE
      NCXPS=IE-IS+1
      NRXPS=JE-JS+1
C
C     SET UP MASK FOR PRESSURE DATA REQUIRED
C
      ASSIGN BITPD,BITP(1;NPEL)
      ASSIGN ISEAD,ID(1;NPEL)
      BITPD=BITPD.AND..NOT.BITPD
      ISEAD=0
      DO 3 JR=JS,JE
      DO 3 IC=IS,IE
      II=(JR-1)*NCXP+IC
      ID(II)=1
    3 CONTINUE
      BITPD=ISEAD.EQ.1
      IPO=Q8SCNT(BITPD)
C
C     COMPUTE ALL PARAMETERS FOR LINEAR INTERPOLATION OF PRESSURES
C     TO SEA MODEL Z-POINTS
C
C     USING CHIV AND PHIU
      DO 39 I=1,NCX
      IMP(I)=1+IFIX((CHIV(I)-CHINM)*DXMI)
   39 CHIM(I)=CHINM+(IMP(I)-1)*DXM
C
```

```
      DO 40 J=1,NRX
      JMP(J)=1+IFIX((PHINM-PHIU(J))*DYMI)
   40 PHIM(J)=PHINM-(JMP(J)-1)*DYM
C
C     CHIM,PHIM CONTAIN LONG,LAT OF PRESSURE POINTS IN SEA MODEL SPACE
C
      AU(1;NCX)=(CHIV(1;NCX)-CHIM(1;NCX))*DXMI
      BU(1;NRX)=(PHIM(1;NRX)-PHIU(1;NRX))*DYMI
C
C     INTERPOLATION FACTORS FOR SEA MODEL Z-POINTS
C
      K=0
      DO 41 J=1,NRX
      B=BU(J)
      DO 41 I=1,NCX
      K=K+1
      AZXX(K)=AU(I)
      BZXX(K)=B
   41 IIMZX(K)=(JMP(J)-JS)*NCXPS+IMP(I)-IS+1
      WRITE(NW,96)
      CALL ARPRIN(DUM,IIMZX,NCX,NRX,1)
C
C     IIMZX IS 1-D ARRAY OF MET POINT NUMBERS FOR EACH SEA MOD Z-POINT
C     LESS 1 TO GIVE CORRECT ADDRESS RELATIVE TO START LOCN IN Q8VXTOV
C
C     NOW SET INTEGER CONSTANTS FOR OUTPUT IN IDM
C
      IDM(1)=IWO
      IDM(2)=NWEL
      IDM(3)=NSPAG
      IDM(4)=NCXMS
      IDM(5)=NRXMS
      IDM(6)=IPO
      IDM(7)=NPEL
      IDM(8)=NSPAP
      IDM(9)=NCXPS
      IDM(10)=NRXPS
C
C     READ IN DATA FOR PATCHES
C     IF NO PATCHES SET NRR=NRX, NCC=NCX, IFX=0, JFX=0
C
      READ(NR,100) NRR,NCC,IFX,JFX,NPATCH
      WRITE(NW,100) NRR,NCC,IFX,JFX,NPATCH
      ISO=NRR*NCC
      IF(NPATCH.EQ.0) GOTO 98
C
C     READ IN DATA FOR PATCH RELOCATION
C
      DO 91 I=1,NPATCH
      READ(NR,100) IXST(I),JXST(I),NCT(I),NRT(I),IF(I),JF(I)
   91 CONTINUE
C
C     READ IN LABELS
C
      READ(NR,101) (LABX(I),I=1,ISX)
  101 FORMAT(20I4)
C
C     RELOCATE PATCHES
```

```
C
      DO 92 I=1,NPATCH
      NCTN=NCT(I)
      NRTN=NRT(I)
      DO 93 IC=1,NCTN
      DO 93 JR=1,NRTN
      IX=(JXST(I)+JR-2)*NCX+IXST(I)+IC-1
      IP=IX+JF(I)*NCX+IF(I)
      IF(LABX(IX).NE.0.AND.LABX(IP).NE.0) WRITE(NW,94) N,IC,JR
   94 FORMAT(///' DATA BEING OVERWRITTEN IN PATCH TRANSFER ',3I5)
      IF(LABX(IX).EQ.0) GOTO 95
      IIMUX(IP)=IIMUX(IX)
      IIMVX(IP)=IIMVX(IX)
      IIMZX(IP)=IIMZX(IX)
      AUXX(IP)=AUXX(IX)
      AVXX(IP)=AVXX(IX)
      AZXX(IP)=AZXX(IX)
      BUXX(IP)=BUXX(IX)
      BVXX(IP)=BVXX(IX)
      BZXX(IP)=BZXX(IX)
   95 CONTINUE
   93 CONTINUE
   92 CONTINUE
C
C     END OF PATCHWORK
C
   98 CONTINUE
      WRITE(NW,96)
   96 FORMAT(1H1)
      CALL ARPRIN(DUM,IIMUX,NCX,NRX,1)
      WRITE(NW,96)
      CALL ARPRIN(DUM,IIMVX,NCX,NRX,1)
      WRITE(NW,96)
      CALL ARPRIN(DUM,IIMZX,NCX,NRX,1)
C
C     EXTRACT RECTANGLE
C
      DO 97 I=1,NCC
      DO 97 J=1,NRR
      II=(J-1)*NCC+I
      IX=(J+JFX-1)*NCX+I+IFX
      IIMU(II)=IIMUX(IX)
      IIMV(II)=IIMVX(IX)
      IIMZ(II)=IIMZX(IX)
      AUX(II)=AUXX(IX)
      AVX(II)=AVXX(IX)
      AZX(II)=AZXX(IX)
      BUX(II)=BUXX(IX)
      BVX(II)=BVXX(IX)
      BZX(II)=BZXX(IX)
   97 CONTINUE
C
C     WRITE DATA TO DISC NWM
C
      CALL Q7BUFOUT(NWM,DAM,55,'SMALL')
      CALL Q7WAIT(NWM,DAM,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,207) ISTAT
  207 FORMAT(10X,'ABNORMAL END OF BUMRI  ISTAT= ',I5)
```

```
      CALL Q7BUFOUT(NWM,BITIM,1,'SMALL')
      CALL Q7WAIT(NWM,BITIM,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,208) ISTAT
  208 FORMAT(10X,'ABNORMAL END OF BUMIT  ISTAT= ',I5)
C
C     PRINT DATA BEING STORED  -   EXCLUDING BIT MASKS
C
      WRITE(NW,120)
  120 FORMAT(1H1,////,'  DATA FOR INTERPOLATION OF MET DATA  ',///)
      WRITE(NW,121) (IDM(I),I=1,10)
  121 FORMAT(5X,20I5)
      WRITE(NW,121) (IIMU(I),I=1,ISO)
      WRITE(NW,121) (IIMV(I),I=1,ISO)
      WRITE(NW,121) (IIMZ(I),I=1,ISO)
      WRITE(NW,122) (AUX(I),I=1,ISO)
      WRITE(NW,122) (AVX(I),I=1,ISO)
      WRITE(NW,122) (AZX(I),I=1,ISO)
      WRITE(NW,122) (BUX(I),I=1,ISO)
      WRITE(NW,122) (BVX(I),I=1,ISO)
      WRITE(NW,122) (BZX(I),I=1,ISO)
  122 FORMAT(5X,20F6.3)
      STOP
      END
      SUBROUTINE ARPRIN(A,IA,NC,NR,IFO)
      DIMENSION A(1),IA(1)
      NW=6
      DO 10 IC=1,999
      IS=(IC-1)*20+1
      IE=IC*20
      IF(IE.GT.NC) IE=NC
      IF(IS.GT.NC) GOTO 12
      IF(IC.NE.1) WRITE(NW,100)
  100 FORMAT(1H1)
      DO 11 J=1,NR
      ISS=(J-1)*NC+IS
      IEE=(J-1)*NC+IE
      IF(IFO.EQ.1) WRITE(NW,101) J,(IA(II),II=ISS,IEE)
  101 FORMAT(1X,I5,5X,20I4)
      IF(IFO.EQ.2) WRITE(NW,102) J,(A(II),II=ISS,IEE)
  102 FORMAT(1X,I5,5X,20F5.0)
   11 CONTINUE
   10 CONTINUE
   12 CONTINUE
      RETURN
      END
```

Input data for program METSET

```
 52   57
  -15.75      62.5
0.008726646 0.005817764
 46   44    7    0    3
  2   26    8   12   42  -25
 17   45   14    8   18   -6
 50   11    8   12   -6   18
   0   0   0   0   0   0   0   0  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1  -1
  -1  -1  -1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0  -1 222 221 221 221 221 221 221 221 201   0 201 221 221 221 221
 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 221 200
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0  -1 212 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212
 111 111 111 111 111 111 111 111 101 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101 111 111 111 111 111
 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 101 110 111 111 111 111 111 111 111 111
 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 101   0 111 111 111 111 111 111 111 111 111 111 111
 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0  -1 212 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  -1 212 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 100 101 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0
   0   0   0 101   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111
 111 111 111 111 111 111 111 111 111 110 110 110 110 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0 111 111
 101   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111
 111 111 100 111 111 100   0   0   0   0 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 101   0   0   0   0   0   0 111 111 111 111 101   0
   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 101   0
 111 101   0   0   0   0 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 101   0   0   0   0 111 111 111 111 111 101   0   0   0   0
   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 100 111 110 100   0
   0   0 110 110 110 110 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 110 111 111 101   0   0   0   0   0   0
   0   0  -1 212 111 111 111 111 111 111 111 101   0 101   0   0   0   0   0   0
   0   0   0 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 100   0 111 111 101   0   0   0   0   0   0   0   0  -1
 212 111 111 111 111 111 111 111 101 111 101   0   0   0   0   0   0   0   0   0
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 110 110 100   0   0 111 111 111 101   0   0   0   0   0   0   0  -1 212 111 111
 111 111 111 111 111 111 111 110 101   0   0   0   0   0   0   0 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0
```

```
   0   0 110 111 111 101   0   0   0   0   0   0   0  -1 212 111 111 111 111 111
 111 111 111 111 111 100   0   0   0   0   0   0 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0
 111 111 101   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111
 111 111 110 101   0   0   0   0   0 110 110 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 111 111 101   0   0   0   0   0 111 111 110 100
   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 111 101   0
 101   0   0   0   0   0   0   0 110 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 111 111 101   0   0   0   0 110 111 100   0   0   0   0   0
   0   0   0   0  -1 212 111 111 111 111 111 111 111 110 110 110 110 101 111 100
   0   0   0   0   0   0 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 101   0   0   0   0 100   0   0   0   0   0   0   0   0   0
   0  -1 212 111 111 111 111 111 111 100   0   0   0   0 111 101   0   0   0   0
   0   0   0 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212
 111 111 111 111 111 101   0   0   0   0   0 110 101   0 111 101   0   0   0   0
 110 110 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111
 110 110 100   0   0   0   0   0   0 111 111 111 101   0   0   0   0   0   0 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 101   0   0
   0   0   0   0   0   0   0   0  -1  -1  -1  -1  -1 211 111 111 101   0   0   0
   0   0   0   0   0 111 111 101 111 111 101   0   0   0   0   0 110 111 111 111
 111 111 111 111 111 111 111 111 111 111 111 111 111 111 100   0   0   0   0   0
   0   0   0   0  -1 221 221 221 221 211 111 111 111 101   0   0   0   0   0   0
   0   0 111 111 111 111 111 101   0   0   0   0   0   0 111 111 111 111 111 111
 111 111 111 111 111 111 111 111 110 110 100   0   0   0   0   0   0   0   0   0
  -1 212 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0 111
 111 111 110 110 100   0   0   0   0   0   0 111 111 111 111 111 111 111 111 111
 111 111 110 110 100   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111
 111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0 111 111 101   0
   0   0   0   0   0   0   0   0 110 110 110 111 111 111 111 111 111 110 100   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111
 111 111 111 111 111 101   0   0   0   0   0   0   0 111 111 111 101   0   0   0
   0   0   0   0   0   0   0   0 111 111 111 111 111 101   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111
 111 111 100   0   0   0   0   0   0 111 111 111 111 100   0   0   0   0   0   0
   0   0   0   0   0 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 100   0
   0   0   0   0   0 111 111 111 111 100   0   0   0   0   0   0   0   0   0   0
   0   0 111 111 111 111 111 100   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 101   0   0   0   0
 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0 111 111
 111 111 111 100   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0  -1 212 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 111 111 111 111 110 100   0   0   0   0   0   0   0 110 110 111 111 111 100
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1
 210 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111 111
 111 100   0   0   0   0   0   0   0   0   0   0 111 110 110 100   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 220 220
 220 220 220 210 111 111 111 111 111 111 111 111 111 111 111 111 111 100   0   0
   0   0   0   0   0   0   0   0 111 101   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1  -1  -1  -1  -1
  -1 212 111 111 111 111 111 111 111 111 111 111 111 100   0   0   0 111 111 111
 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111
 111 111 111 111 111 111 111 111 111 101   0 111 111 111 111 111 111 111 111 111
 111 111 110 100   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111
```

```
111 111 111 111 111 111 111 111 111 111 111 111 111 111 110 111 111 111 100   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111
111 111 111 111 111 111 111 111 111 111 101   0 110 110 100   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0  -1 212 111 111 111 111 111 111 111 111 111 111
111 111 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0  -1 210 111 111 111 111 111 111 111 111 111 111 111 111 111
110 110 110 110 110 100   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0  -1 220 210 111 111 111 111 111 111 111 111 111 111 101   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0  -1  -1 220 220 220 210 111 111 111 111 111 111 101   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
 -1  -1  -1  -1 210 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0  -1 220 220 210 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1
 -1  -1 210 111 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1
210 111 111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 210 111
111 111 111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  -1 210 111 111 111
111 111 101   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0  -1 220 220 220 220 220 200
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0   0   0   0   0   0   0   0  -1  -1  -1  -1  -1  -1   0   0   0
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  0   0   0   0
```

```
      PROGRAM GESMOD(UNIT5[,,4]=CPCDS,UNIT6=OUTPUT,UNIT9[,,4]=CPCF1,
     1UNIT10[,,4]=CPTID1,UNIT11[,,4]=CPSUR1,UNIT12[,,4]=CPTID2,
     2UNIT13[,,4]=CPSUR2,UNIT14[,,4]=CSPID1,UNIT15[,,4]=CPSMD3,
     3UNIT16[,,4]=WINDFMF,UNIT17[,,4]=FMFCSP,UNIT23[,,4]=CSPBIT,
     4UNIT18[,,4]=CPTAB1,UNIT19[,,4]=CSPRES)
C***************************************************************
C*                                                             *
C*      MODIFIED 30/6/81  SURGE INPUT CURRENT SWITCH INCLUDED  *
C*      DEVICE NUMBERS FOR TABLE,SEA MOD DATA AND 24HR DATA    *
C*      CYBER VERSION ********* 15/07/82                       *
C*      MODULAR VERSION ******* 02/08/82                       *
C*      TIDE +SURGE MODEL ***** 10/08/82                       *
C*      WITH MET PROCESSING *** 16/08/82                       *
C*      AND INITIALISATION **** 18/08/82                       *
C*      PATCHED VERSION ******* 08/10/82                       *
C*      INCLUDES DRYING ******* 12/01/83                       *
C*      AND I/O OF RESIDUALS ** 12/01/83                       *
C*                                                             *
C***************************************************************
      DIMENSION H(3072),IZOB(150),IUOB(150),IVOB(150),VPLU(4),FTC(4),
     C CORUA(3072),CORVA(3072),CSPUAI(3072),CSPVA(3072)
      DIMENSION IZOBN(150),IZOBS(150),IZOBW(150),IZOBE(150)
      DIMENSION ATC(4,3),ICTC(4,2),BTC(4)
      DIMENSION JTIM(5,4),IHLAP(4)
      DIMENSION BITZ(3072),BITU(3072),BITV(3072),BITUX(3072),BITVX(3072)
      DIMENSION BITZO(3072)
      DIMENSION ZT(3072),UT(3072),VT(3072),ZS(3072),US(3072),VS(3072)
      DIMENSION NPIN(3,100),NPOT(3,100)
      BIT BITUX,BITVX
      BIT BITIO
      BIT BITH,BITHD
      BIT BITZ,BITZD,BITU,BITUD,BITV,BITVD
      BIT BITZO,BITZOD,BITIM,BITW BITP,BITWD,BITPD
      DESCRIPTOR BITZOD
      CHARACTER*4 IZZ(20),ITIT(20),ITIP(60)
      COMMON/BUFIT/BITIO(32768)
      COMMON/BUFRI/DATA(17408)
      COMMON/BUFIO/ZIO(1536,3)
      COMMON/HEADW/WHEAD(512)
      COMMON/HEADP/PHEAD(512)
      COMMON/BUFWI/DATAWH(8704)
      COMMON/BUFPH/DATAPH(4352)
      COMMON/BUMIT/BITIM(32768)
      COMMON/BUMRI/DAM(28160)
      COMMON/COM1/Z(3072),U(3072),V(3072),ZZ(3072),UU(3072),VV(3072)
      COMMON/COM2/NW,N,NUM(60),NCC,NRR,NCON,ITSM
      COMMON/COM3/ZETT(60,49),ZETS(60,49),ZETR(60,49),IZET(60)
      COMMON/COM4/DT,FR,DCRIT,ZCRIT,GC,RE,DTRX,DTRY,GDTRX,GDTRY,
     C              DTRO,FRDT,ITOV,ION,NTPH
      COMMON/COM5/LLMI,LMI,ISWSC,TIME,IPCL,LO
      COMMON/COM6/ZB1(150),ZB2(150),UB1(150),UB2(150),VB1(150),VB2(150),
     C    Z1(150,4),Z2(150,4),U1(150,4),U2(150,4),V1(150,4),V2(150,4),
     C    SIG(4),ZINT(150),ZINS(150),P1(3072),P2(3072),F1(3072),F2(3072),
     C    G1(3072),G2(3072)
      EQUIVALENCE (BITZ(1),BITIO(1)),(BITU(1),BITIO(3073)),
     C              (BITV(1),BITIO(6145)),(BITUX(1),BITIO(9217)),
     C              (BITVX(1),BITIO(12289)),(BITZO(1),BITIO(15361))
      EQUIVALENCE (DATA(1),H(1)),(DATA(3073),CORUA(1)),
     C (DATA(6145),CORVA(1)),(DATA(9217),CSPUAI(1)),
     C (DATA(12289),CSPVA(1)),(DATA(15361),IZOB(1)),
     C (DATA(15511),IUOB(1)),(DATA(15661),IVOB(1)),
     C (DATA(15811),IZOBN(1)),(DATA(15961),IZOBS(1)),
     C (DATA(16111),IZOBW(1)),(DATA(16261),IZOBE(1)),
     C (NNRR,DATA(16500)),(NNCC,DATA(16501)),(ITOT,DATA(16502)),
     C (IINZ,DATA(16503)),(IMEU,DATA(16504)),(IMEV,DATA(16505)),
     C (IINUX,DATA(16506)),(IINVY,DATA(16507)),(IOBZ,DATA(16508)),
     C (IOBU,DATA(16509)),(IOBV,DATA(16510)),(IOBZN,DATA(16511)),
     C (IOBZS,DATA(16512)),(IOBZW,DATA(16513)),(IOBZE,DATA(16514)),
     C (DX,DATA(16515)),(DY,DATA(16516))
      EQUIVALENCE (NTZ,DATA(16517)),(NTU,DATA(16518)),
     C   (NTV,DATA(16519)),(NPIN(1,1),DATA(16520)),
     C   (NPOT(1,1),DATA(16820))
      NR = 5
      NW = 6
      NRWC=9
      NRM=14
      NRS=15
      NRW=16
      NRP=17
      NWTT=18
      NRES=20
      NBOE=23
C
C     NR - READ FROM CARDS
C     NW - PRINT OUT
C     NRWC- READ OR WRITE TO FORECAST CONTROL FILE
C     NRM- READ CONTROL DATA FOR MET PROCESSING FROM DISC
C     NRS- READ FROM SEA MODEL DATA FILE ON DISC
C     NRW- READ FROM WIND INPUT DATA FILE ON DISC
C     NRP- READ FROM PRESSURE INPUT DATA FILE ON DISC
C     NWTT- WRITE ELEVATION TABLES TO DISC
C     NBOE - READ TIDAL INPUT DATA FROM DISC
C
C     READ DATA FROM FORECAST CONTROL FILE
C     SETS FILES FOR INITIAL DATA INPUT
C
      READ(NRWC) NRIT,NRIS,NWFT,NWFS
      REWIND NRWC
C     NRIT- READ FROM INITIAL VALUE FILE ON DISC (TIDE)
C     NRIS- READ FROM INITIAL VALUE FILE ON DISC (TIDE+SURGE)
C     NWFT- WRITE COMPACT ARRAYS TO FILE ON DISC (TIDE)
C     NWFS- WRITE COMPACT ARRAYS TO FILE ON DISC (TIDE+SURGE)
C
C     READ GENERAL INPUT DATA FROM CARDS
C
      READ(NR,101) IZZ
      READ(NR,102) DT,FR
      READ(NR,102) DCRIT,ZCRIT
      READ(NR,113) ITS,LHR,LPR,LET,LPUN,IPCL,LGDG,LGDGE,ILAG,ISWSC,
     &IPCW,NPRTS,IPCP,NWFR,IROB
      READ(NR,103) JRUN,LEG
      READ(NR,101) (ITIT(J),J=1,20)
      READ(NR,113) (IZET(J),J=1,NPRTS)
      READ(NR,101) (ITIP(J),J=1,NPRTS)
  101 FORMAT(20A4)
  102 FORMAT(F10.1,F10.4)
  103 FORMAT(8I10)
  113 FORMAT(16I5)
C
C     NTPH=NO. OF TIME STEPS PER HOUR
C
      NTPH=IFIX(3600./DT+0.01)
C
C     READ INITIAL DATA TIME FROM MET INPUT
C
      CALL Q7BUFIN(NRW,WHEAD,1,'SMALL')
      CALL Q7WAIT(NRW,WHEAD,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,109) ISTAT
  109 FORMAT(10X,'ABNORMAL END TO WHEAD, STAT=',I5)
      ITIME=0
      IHR=WHEAD(5)
      IDAY=WHEAD(6)
      IMNTH=WHEAD(7)
      IYR=WHEAD(8)
      NSPW=(2*WHEAD(3)+1023)/1024
C
      CALL Q7BUFIN(NRP,PHEAD,1,'SMALL')
      CALL Q7WAIT(NRP,PHEAD,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,110) ISTAT
  110 FORMAT(10X,'ABNORMAL END TO PHEAD, STAT=',I5)
      NSPP=(PHEAD(3)+1023)/1024
      JTIM(1,1)=0
      JTIM(1,2)=0
      DO 200 I=2,5
      II=I+3
      JTIM(I,1)=WHEAD(II)
  200 JTIM(I,2)=PHEAD(II)
C     START TIMES OF WIND AND PRESSURE DATA
C
      IHS=IHR+ITIME
      IF(ILAG.EQ.0) GOTO 53
      IHS=IHS+ILAG
   52 CONTINUE
      IF(IHS.LT.24) GOTO 51
      IHS=IHS-24
      IDAY=IDAY+1
      GOTO 52
   51 CONTINUE
C     ***********************************************************
      CALL VDAY(IDAY,IMNTH,IYR,IVDY)
C     COMPUTES DAY NUMBER IN YEAR
      CALL DATEB(IVDY,IYR,IDAY,IMNTH,IYEAR)
      IYR=IYEAR
C     COMPUTES REVISED AND CORRECTED DATE
C     ***********************************************************
C     INITIAL TIME OF SEA MODEL FORECAST
   53 CONTINUE
C
C     PRINT TITLE AND GENERAL INPUT DATA
C
      WRITE(NW,104) JRUN,LEG
  104 FORMAT(1H1/////////' SEA MODEL FOR TIDES AND STORM SURGES'/
     &2X,'RUN',I5/2X,'LEG',I5)
      WRITE(NW,119) IZZ
  119 FORMAT(1H0,20A4)
      WRITE(NW,105) IHS,IDAY,IMNTH,IYR
  105 FORMAT(1H0,'INITIAL DATA TIME ',I3,' HOURS GMT ON',I3,1H/,I2,1H/,
     &I4)
      WRITE(NW,106) DT,FR,ITS,LHR,NTPH,LPR,LET,LPUN,IPCL,IPCW,IPCP,NPRTS
  106 FORMAT(1H0,'INPUT DATA'/
     &2X,'   DT=',F10.1,' TIMESTEP IN SECONDS'/
     &2X,'   FR=',F10.4,' FRICTIONAL COEFFICIENT'/
     &2X,'  ITS=',I10,' COMPUTATION CONTROL PARAMETER  1 FOR SURGE  -1
     &FOR TIDE  0 FOR BOTH'/
     &2X,'  LHR=',I10,' NUMBER OF HOURS IN THIS LEG'/
     &2X,' NTPH=',I10,' NUMBER OF TIMESTEPS PER HOUR'/
     &2X,'  LPR=',I10,' NUMBER OF HOURS BETWEEN EACH PRINTOUT'/
     &2X,'  LET=',I10,' NUMBER OF HOURS BETWEEN ENTRIES IN TABLES'/
     &2X,' LPUN=',I10,' CONTROL PARAMETER FOR STORAGE OF TABLES'/
     &2X,' IPCL=',I10,' PRINTOUT CONTROL PARAMETER'/
     &2X,' IPCW=',I10,' MET PRINTOUT CONTROL PARAMETER  O/P EVERY IPCW H
     &RS  -VE TO SUPRESS'/
     &2X,' IPCP=',I10,' CONTROLS PARAMETERS TO BE PRINTED 0=RESIDUAL ONL
     &Y  1= T+S,T,S '/
     &2X,'NPRTS=',I10,' NUMBER OF PORTS IN TABLE')
      WRITE(NW,107) LGDG,LGDGE,ILAG,ISWSC,DCRIT,ZCRIT
  107 FORMAT(
     &2X,' LGDG=',I10,' NUMBER OF HOURS BETWEEN DATA STORAGE IN GENERATI
     &ON DATA GROUP'/
     &2X,'LGDGE=',I10,' LAST HOUR AT WHICH DATA WRITTEN IN GENER. DATA G
     &ROUP'/
     &2X,' ILAG=',I10,' TIME LAG HRS BETWEEN INITIAL DATA TIME FOR THIS
     &RUN AND OF CURRENT MET DATA'/
     &2X,'ISWSC=',I10,' CONTROL PARAMETER FOR INPUT OF SURGE CURRENTS'/
     &2X,'DCRIT=',F10.1,' CRITICAL DEPTH FOR DRYING TEST'/
     &2X,'ZCRIT=',F10.4,' CRITICAL Z-DIFFERENCE FOR DRYING')
      WRITE(NW,114) (IZET(J),J=1,NPRTS)
  114 FORMAT(1H0,34H ELEVATION TABLES GIVEN FOR POINTS/(2X,10I8))
      WRITE(NW,116) NRIT,NRIS,NWFT,NWFS,NWFR,IROB
  116 FORMAT('    NRIT=',I5,'    NRIS=',I5,'    NWFT=',I5,'    NWFS=',I5,/
     &'    NWFR=',I5,'    IF=0, NO RESIDUAL O/P'/
     C' IROB=',I5,5X,'BOUNDARY RESIDUAL PARAMETER, 0 FOR HYDROSTATIC, 1
     CFOR PRESCRIBED INPUT')
C
C     ELEVATION TABLES FOR 60 POINTS ARE PROVIDED EACH WITH A MAXIMUM
C     OF 49 ENTRIES IN A RUN
C
      ZETT(1,1;60*49)=1.0E10
      ZETS(1,1;60*49)=1.0E10
C
C     READ SEA MODEL DATA FROM DISC
C
C     NRR= NUMBER OF ROWS
C     NCC= NUMBER OF COLUMNS
C     ITOT=TOTAL NUMBER OF POINTS
C     IINZ=NUMBER OF INTERNAL Z-POINTS
C     IOBZ=NUMBER OF OPEN BOUNDARY Z-POINTS
C     DX   GRID INCREMENT IN X-DIRECTION
C     DY   GRID INCREMENT IN Y-DIRECTION
C     IMEU=NUMBER OF U-MET DATA POINTS
C     IMEV=NUMBER OF V-MET DATA POINTS
C     IOBU=NUMBER OF OPEN BOUNDARY U-POINTS
```

```
C     IOBV=NUMBER OF OPEN BOUNDARY V-POINTS
      CALL Q7BUFIN(NRS,DATA,34,'SMALL')
      CALL Q7WAIT(NRS,DATA,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,141) ISTAT
  141 FORMAT(10X,' ABNORMAL END OF BUFRI  STAT=',I5)
      CALL Q7BUFIN(NRS,BITIO,1,'SMALL')
      CALL Q7WAIT(NRS,BITIO,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,140) ISTAT
  140 FORMAT(10X,' ABNORMAL END OF BUFIN  STAT=',I5)
C
C     READ CONTROL DATA FOR MET PROCESSING FROM DISC
C
      CALL Q7BUFIN(NRM,DAM,55,'SMALL')
      CALL Q7WAIT(NRM,DAM,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,142) ISTAT
  142 FORMAT(10X,' ABNORMAL END OF BUMRI  ISTAT= ',I5)
      CALL Q7BUFIN(NRM,BITIM,1,'SMALL')
      CALL Q7WAIT(NRM,BITIM,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,143) ISTAT
  143 FORMAT(10X,' ABNORMAL END OF BUMIT  ISTAT= ',I5)
C
C     SET FIXED PARAMETERS
C
      NRR=NNRR
      NCC=NNCC
      IINT=MAX0(NRR,NCC)
C     IINT=NUMBER OF INTEGER ROW OR COLUMN LABELS
      N=NCC
C     N    NUMBER OF GRID POINTS ALONG A ROW
      GC=9.81
C     GC   GRAVITATIONAL CONSTANT
      RE=6.37E6
C     RE   RADIUS OF THE EARTH
      RO=1025.0
C     RO   DENSITY OF SEA WATER
      ROA=1.25
C     ROA  DENSITY OF AIR
      PMEAN=1012.0
C     PMEAN  MEAN ATMOSPHERIC PRESSURE MB
      DTRX=DT/(RE*DX)
      DTRY=DT/(RE*DY)
      GDTRX=GC*DTRX
      GDTRY=GC*DTRY
      DTRO=DT/RO
      FRDT=FR*DT
      ITOV=ITOT-2*N
      ION=1536
C
C     INTRODUCE AN INTEGER ARRAY FOR ROW AND COLUMN DESIGNATION
C     IN PRINTED OUTPUT
C
      DO 8 J=1,IINT
      NUM(J)=J
    8 CONTINUE
C
C      INITIALISE TIDE AND TIDE+SURGE ARRAYS
C
C
```

```
C     INITIALISE ZZ UU VV TO ZERO
C
      ZZ(1;ITOT)=0.0
      UU(1;ITOT)=0.0
      VV(1;ITOT)=0.0
C
C     READ INITIAL VALUE DATA FROM DISC FOR TIDE AND TIDE+SURGE
C     WITH CHECK ON RUN TIMES TO MATCH INPUT ARRAYS
C
C     READ IN SET OF TIDE ARRAYS
C
  996 CONTINUE
      CALL Q7BUFIN(NRIT,ZIO,9,'SMALL')
      CALL O7WAIT(NRIT,ZIO,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,120) ISTAT
  120 FORMAT(10X,' ABNORMAL END TO BUFIOT IN, STAT=',I5)
      IF(ISTAT.EQ.1) GOTO 999
C     END OF TIDE ARRAY INPUT FILE REACHED
      ZT(1;ITOT)=Q8VXPND(ZIO(1,1;IINZ),BITZ(1;ITOT);ZT(1;ITOT))
      UT(1;ITOT)=Q8VXPND(ZIO(1,2;IINUX),BITUX(1;ITOT);UT(1;ITOT))
      VT(1;ITOT)=Q8VXPND(ZIO(1,3;IINVX),BITVX(1;ITOT);VT(1;ITOT))
      DO 202 I=1,5
      II=I+ION-5
  202 JTIM(I,3)=ZIO(II,3)
C     START TIME OF TIDE DATA
C
  999 CONTINUE
C
C     READ IN SET OF TIDE + SURGE ARRAYS
C
      CALL Q7BUFIN(NRIS,ZIO,9,'SMALL')
      CALL Q7WAIT(NRIS,ZIO,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,221) ISTAT
  221 FORMAT(10X,' ABNORMAL END TO BUFIOS IN, STAT=',I5)
      IF(ISTAT.EQ.1) GOTO 998
C     END OF TIDE+SURGE ARRAY INPUT FILE REACHED
      ZS(1;ITOT)=Q8VXPND(ZIO(1,1;IINZ),BITZ(1;ITOT);ZS(1;ITOT))
      US(1;ITOT)=Q8VXPND(ZIO(1,2;IINUX),BITUX(1;ITOT);US(1;ITOT))
      VS(1;ITOT)=Q8VXPND(ZIO(1,3;IINVX),BITVX(1;ITOT);VS(1;ITOT))
      DO 203 I=1,5
      II=I+ION-5
  203 JTIM(I,4)=ZIO(II,3)
C     TIME OF TIDE + SURGE DATA
      CALL STARTI(NW,JTIM,IHLAP)
      ITWIND=IHLAP(1)
      ITPRES=IHLAP(2)
      ITTIDE=IHLAP(3)
      ITSURG=IHLAP(4)
      IF(ITSURG.GT.ITWIND) WRITE(NW,300)
  300 FORMAT(10X,' FORECAST ALREADY RUN - STOP NOW '//)
      IF(ITSURG.GT.ITWIND) GOTO 990
      IF(ITSURG.EQ.ITWIND) GOTO 997
C     PREVIOUS FORECAST RAN NORMALLY - CONTINUE
      IF(ITSURG.LT.ITWIND) WRITE(NW,301)
  301 FORMAT(10X,' FORECAST MISSED - TRY NEXT SET OF SEA MOD DATA ')
      IF(ITSURG.LT.ITWIND) GOTO 996
  998 CONTINUE
C     NO INITIAL DATA FOUND - COLD START
```

```
      ZT(1;ITOT)=0.0
      UT(1;ITOT)=0.0
      VT(1;ITOT)=0.0
      ZS(1;ITOT)=0.0
      US(1;ITOT)=0.0
      VS(1;ITOT)=0.0
      WRITE(NW,302) ITIME,IHR,IDAY,IMNTH,IYR
  302 FORMAT(10X,' COLD START AT ',5I5)
  997 CONTINUE
      LO=0
C
C     INITIALISATION NOW COMPLETE
C
C
C     FOR SURGES READ AND PRINT RELEVENT DATA
C
      READ(NR,113) LMI
      WRITE(NW,121) LMI
  121 FORMAT(1H0,' INPUT DATA FOR SURGE'/
     &2X,' LMI=',I10,' NUMBER OF HOURS BETWEEN MET DATA INPUT')
C
C     READ WIND STRESS PARAMETERS - LINEAR VARIATION OF DRAG COEFF
C     WITH WIND SPEED ASSUMED
C
      READ(NR,115) A0,A1
  115 FORMAT(2E12.4)
      WRITE(NW,122) A0,A1
  122 FORMAT(2X,' A0=',E12.4,' A1=',E12.4,' DRAG COEFF PARAMETERS')
C
C     READ INITIAL SET OF WINDS AND PRESSURES
C
      CALL Q7BUFIN(NRW,DATAWH,NSPW,'SMALL')
      CALL Q7WAIT(NRW,DATAWH,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,125) ISTAT
  125 FORMAT(10X,'ABNORMAL END TO BUFWI, STAT=',I5)
      CALL Q7BUFIN(NRP,DATAPH,NSPP,'SMALL')
      CALL Q7WAIT(NRP,DATAPH,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,226) ISTAT
  226 FORMAT(10X,'ABNORMAL END TO BUFPR, STAT=',I5)
C
C     EVALUATE STRESSES, AND PRESSURES AND EXTRACT BOUNDARY SURGE
C     INPUTS FOR THE SEA MODEL ASSUMING HYDROSTATIC LAW
C
      LOOP=0
      CALL METPROC(ITOT,A0,A1,ROA,RO,GC,PMEAN,LOOP,IPCW)
      IF(IROB.NE.0) GOTO 42
      ASSIGN BITZOD,BITZO(1;ITOT)
      ZB2(1;IOBZ)=Q8VCMPRS(P2(1;ITOT),BITZOD;ZB2(1;IOBZ))
      UB2(1;IOBU)=0.
      VB2(1;IOBV)=0.
      GOTO 43
   42 READ(NRES,END=43)
      READ(NRES) (ZB2(I),I=1,IOBZ)
      READ(NRES) (UB2(I),I=1,IOBU)
      READ(NRES) (VB2(I),I=1,IOBV)
   43 CONTINUE
C
C     RESET MET COUNTER
```

```
C
      LLMI=LMI
C
C     FOR TIDES READ AND PRINT RELEVENT DATA
C
      IF(ITS) 45,45,47
   45 CONTINUE
C
C     CALCULATE DAY NUMBER AND S,H,P,N
C
C     **********************************
      CALL VDAY(IDAY,IMNTH,IYR,IVDY)
C     COMPUTES DAY NUMBER IN YEAR
C     **********************************
      CALL SHPN(IYR,IVDY,SSS,HHH,PPP,CN,PPPP)
      CN=3.14159265*CN/180.
      W1=COS(CN)
      W2=COS(2.*CN)
      W4=SIN(CN)
      READ(NBOE) NCON
      WRITE(NW,123) NCON
  123 FORMAT(1H0,21H INPUT DATA FOR TIDES/
     &2X,6H NCON=,I10,46H NUMBER OF TIDAL CONSTITUENTS INPUT  MAXIMUM 4)
C
C     READ DATA FOR EACH CONSTITUENT
C
      DO 46 ICON=1,NCON
      READ(NBOE) SIG(ICON)
      READ(NBOE) (ATC(ICON,J),J=1,3)
      READ(NBOE) BTC(ICON)
      READ(NBOE) (ICTC(ICON,J),J=1,2)
      READ(NBOE) (Z1(J,ICON),J=1,IOBZ)
      READ(NBOE) (Z2(J,ICON),J=1,IOBZ)
      READ(NBOE) (U1(J,ICON),J=1,IOBU)
      READ(NBOE) (U2(J,ICON),J=1,IOBU)
      READ(NBOE) (V1(J,ICON),J=1,IOBV)
      READ(NBOE) (V2(J,ICON),J=1,IOBV)
      FTC(ICON)=ATC(ICON,1)+ATC(ICON,2)*W1+ATC(ICON,3)*W2
      UTC=BTC(ICON)*W4
      VTC=ICTC(ICON,1)*SSS+ICTC(ICON,2)*HHH+IHS*SIG(ICON)
      VPLU(ICON)=UTC+VTC
      WRITE(NW,126) ICON,SIG(ICON)
      WRITE(NW,129) VPLU(ICON),FTC(ICON)
      ARG=3.14159265*VPLU(ICON)/180.0
      A=FTC(ICON)*COS(ARG)
      B=FTC(ICON)*SIN(ARG)
      ZINT(1;IOBZ)=A*Z1(1,ICON;IOBZ)+B*Z2(1,ICON;IOBZ)
      ZINS(1;IOBZ)=A*Z2(1,ICON;IOBZ)-B*Z1(1,ICON;IOBZ)
      Z1(1,ICON;IOBZ)=ZINT(1;IOBZ)
      Z2(1,ICON;IOBZ)=ZINS(1;IOBZ)
      ZINT(1;IOBU)=A*U1(1,ICON;IOBU)+B*U2(1,ICON;IOBU)
      ZINS(1;IOBU)=A*U2(1,ICON;IOBU)-B*U1(1,ICON;IOBU)
      U1(1,ICON;IOBU)=ZINT(1;IOBU)
      U2(1,ICON;IOBU)=ZINS(1;IOBU)
      ZINT(1;IOBV)=A*V1(1,ICON;IOBV)+B*V2(1,ICON;IOBV)
      ZINS(1;IOBV)=A*V2(1,ICON;IOBV)-B*V1(1,ICON;IOBV)
      V1(1,ICON;IOBV)=ZINT(1;IOBV)
      V2(1,ICON;IOBV)=ZINS(1;IOBV)
   46 CONTINUE
  126 FORMAT(1H0,30H  ANGULAR SPEED OF CONSTITUENT,I3,3H  =,F10.7,18H  D
     &EGREES PER HOUR/)
  129 FORMAT(17H  TIDAL CONSTANTS/6H  V+U=,F10.4,9H  DEGREES/6H    F=,F1
     &0.6/)
C
   47 CONTINUE
C
C
C     PRINT INITIAL VALUES
C
      TIME=LO
      ITIM=0
      IF(IPCL.EQ.0) GOTO 700
      Z(1;ITOT)=ZS(1;ITOT)-ZT(1;ITOT)
      U(1;ITOT)=US(1;ITOT)-UT(1;ITOT)
      V(1;ITOT)=VS(1;ITOT)-VT(1;ITOT)
      CALL PARFO(ITOT)
      IF(IPCP.EQ.0) GOTO 701
      Z(1;ITOT)=ZS(1;ITOT)
      U(1;ITOT)=US(1;ITOT)
      V(1;ITOT)=VS(1;ITOT)
      CALL PARFO(ITOT)
      Z(1;ITOT)=ZT(1;ITOT)
      U(1;ITOT)=UT(1;ITOT)
      V(1;ITOT)=VT(1;ITOT)
      CALL PARFO(ITOT)
  701 CONTINUE
  700 CONTINUE
      LLPR=LPR
C
C     SET INITIAL VALUES IN ELEVATION TABLES
C
      JET=1
C     JET COUNTS THE NUMBER OF ENTRIES IN THE TABLES
      DO 62 I=1,NPRTS
      IET=IZET(I)
      ZETT(I,JET)=ZT(IET)
      ZETS(I,JET)=ZS(IET)
   62 CONTINUE
      LLET=LET
C
C     OUTPUT INITIAL RESIDUAL ARRAYS IF REQUIRED
C
      IF(NWFR) 31,32,31
   31 CONTINUE
      TIME=LO
      TT=TIME+0.01
      ITIM=TT
      ZIO(1,1;3*ION)=0.
      Z(1;ITOT)=ZS(1;ITOT)-ZT(1;ITOT)
      U(1;ITOT)=US(1;ITOT)-UT(1;ITOT)
      V(1;ITOT)=VS(1;ITOT)-VT(1;ITOT)
      ZIO(1,1;IINZ)=Q8VCMPRS(Z(1;ITOT),BITZ(1;ITOT);ZIO(1,1;IINZ))
      ZIO(1,2;IINZ)=Q8VCMPRS(U(1;ITOT),BITZ(1;ITOT);ZIO(1,2;IINZ))
      ZIO(1,3;IINZ)=Q8VCMPRS(V(1;ITOT),BITZ(1;ITOT);ZIO(1,3;IINZ))
      ZIO(ION-4,3)=ITIM
      ZIO(ION-3,3)=IHS
      ZIO(ION-2,3)=IDAY
      ZIO(ION-1,3)=IMNTH
      ZIO(ION,3)=IYR
      CALL Q7BUFOUT(NWFR,ZIO,9,'SMALL')
   32 CONTINUE
C
C     CHECK RESIDUAL O/P COMPLETE
C
      IF(NWFR) 33,34,33
   33 CONTINUE
      CALL Q7WAIT(NWFR,ZIO,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,716) ISTAT,LOOP
   34 CONTINUE
C
C
C     *************************
C     THE MAIN LOOP STARTS HERE
C     *************************
C
      LLGDG=LGDG
C
      DO 9 LOOP=1,LHR
C
      LO=LO+1
      LLPR=LLPR-1
      LLGDG=LLGDG-1
      LLMI=LLMI-1
      LLET=LLET-1
      IF(LLMI) 61,60,61
   60 CONTINUE
C
C     INPUT NEW MET DATA FROM DISC
C
      CALL Q7BUFIN(NRW,DATAWH,NSPW,'SMALL')
      CALL Q7BUFIN(NRP,DATAPH,NSPP,'SMALL')
C
C     UPDATE MET ARRAYS
C
      P1(1;ITOT)=P2(1;ITOT)
      F1(1;ITOT)=F2(1;ITOT)
      G1(1;ITOT)=G2(1;ITOT)
      ZB1(1;IOBZ)=ZB2(1;IOBZ)
      UB1(1;IOBU)=UB2(1;IOBU)
      VB1(1;IOBV)=VB2(1;IOBV)
   61 CONTINUE
C
C     INTEGRATE FOR TIDES
C
      Z(1;ITOT)=ZT(1;ITOT)
      U(1;ITOT)=UT(1;ITOT)
      V(1;ITOT)=VT(1;ITOT)
C
      ITSM=0
      CALL GESMEQ
C
      ZT(1;ITOT)=Z(1;ITOT)
      UT(1;ITOT)=U(1;ITOT)
      VT(1;ITOT)=V(1;ITOT)
```

```
C
C     CHECK MET INPUT COMPLETE
C
      CALL Q7WAIT(NRW,DATAWH,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,111) ISTAT,LOOP
  111 FORMAT(10X,'ABNORMAL END TO BUFWI, STAT=',I5,'  LOOP=',I5)
      CALL Q7WAIT(NRP,DATAPH,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,112) ISTAT,LOOP
  112 FORMAT(10X,'ABNORMAL END TO BUFPR, STAT=',I5,'  LOOP=',I5)
C
C     OUTPUT TIDE ARRAYS IF REQUIRED
C
      IF(LO.GT.LGDGE) GOTO 401
      IF(LLGDG) 400,400,401
  400 CONTINUE
      TIME=LO
      TT=TIME+0.01
      ITIM=TT
      ZIO(1,1;3*ION)=0.
      ZIO(1,1;IINZ)=Q8VCMPRS(ZT(1;ITOT),BITZ(1;ITOT);ZIO(1,1;IINZ))
      ZIO(1,2;IINUX)=Q8VCMPRS(UT(1;ITOT),BITUX(1;ITOT);ZIO(1,2;IINUX))
      ZIO(1,3;IINVX)=Q8VCMPRS(VT(1;ITOT),BITVX(1;ITOT);ZIO(1,3;IINVX))
      ZIO(ION-4,3)=ITIM
      ZIO(ION-3,3)=IHS
      ZIO(ION-2,3)=IDAY
      ZIO(ION-1,3)=IMNTH
      ZIO(ION,3)=IYR
      CALL Q7BUFOUT(NWFT,ZIO,9,'SMALL')
  401 CONTINUE
C
C     FILL TIDE TABLES
C
      IF(LLET) 404,402,404
  402 CONTINUE
      JET=JET+1
      DO 403 I=1,NPRTS
      IET=IZET(I)
      ZETT(I,JET)=ZT(IET)
  403 CONTINUE
  404 CONTINUE
C
C     PERFORM MET PROCESSING
C
      IF(LLMI) 65,64,65
   64 CONTINUE
      CALL METPROC(ITOT,A0,A1,ROA,RO,GC,PMEAN,LOOP,IPCW)
      IF(IROB.NE.0) GOTO 35
      ZB2(1;IOBZ)=Q8VCMPRS(P2(1;ITOT),BITZOD;ZB2(1;IOBZ))
      UB2(1;IOBU)=0.
      VB2(1;IOBV)=0.
      GOTO 36
   35 READ(NRES,END=36)
      READ(NRES) (ZB2(I),I=1,IOBZ)
      READ(NRES) (UB2(I),I=1,IOBU)
      READ(NRES) (VB2(I),I=1,IOBV)
   36 CONTINUE
      LLMI=LMI
   65 CONTINUE
```

```
C
C     CHECK TIDE O/P COMPLETE
C
      IF(LO.GT.LGDGE) GOTO 405
      IF(LLGDG)406,406,405
  406 CONTINUE
      CALL Q7WAIT(NWFT,ZIO,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,407) ISTAT,LOOP
  407 FORMAT(10X,'ABNORMAL END TO BUFIO OUT T, STAT=',I5,'  LOOP=',I5)
  405 CONTINUE
C
C     INTEGRATE FOR TIDE + SURGE
C
      Z(1;ITOT)=ZS(1;ITOT)
      U(1;ITOT)=US(1;ITOT)
      V(1;ITOT)=VS(1;ITOT)
C
      ITSM=1
      CALL GESMEQ
C
      ZS(1;ITOT)=Z(1;ITOT)
      US(1;ITOT)=U(1;ITOT)
      VS(1;ITOT)=V(1;ITOT)
C
C     OUTPUT ARRAYS FOR TIDE + SURGE IF REQUIRED
C
      IF(LO.GT.LGDGE) GOTO 411
      IF(LLGDG) 410,410,411
  410 CONTINUE
      TIME=LO
      TT=TIME+0.01
      ITIM=TT
      ZIO(1,1;3*ION)=0.
      ZIO(1,1;IINZ)=Q8VCMPRS(ZS(1;ITOT),BITZ(1;ITOT);ZIO(1,1;IINZ))
      ZIO(1,2;IINUX)=Q8VCMPRS(US(1;ITOT),BITUX(1;ITOT);ZIO(1,2;IINUX))
      ZIO(1,3;IINVX)=Q8VCMPRS(VS(1;ITOT),BITVX(1;ITOT);ZIO(1,3;IINVX))
      ZIO(ION-4,3)=ITIM
      ZIO(ION-3,3)=IHS
      ZIO(ION-2,3)=IDAY
      ZIO(ION-1,3)=IMNTH
      ZIO(ION,3)=IYR
      CALL Q7BUFOUT(NWFS,ZIO,9,'SMALL')
  411 CONTINUE
C
C     FILL TIDE + SURGE TABLES
C
      IF(LLET) 414,412,414
  412 CONTINUE
      DO 413 I=1,NPRTS
      IET=IZET(I)
      ZETS(I,JET)=ZS(IET)
  413 CONTINUE
      LLET=LET
  414 CONTINUE
C
C     CHECK TIDE + SURGE O/P COMPLETE
C
      IF(LO.GT.LGDGE) GOTO 415
      IF(LLGDG)416,416,415
  416 CONTINUE
      CALL Q7WAIT(NWFS,ZIO,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,417) ISTAT,LOOP
  417 FORMAT(10X,'ABNORMAL END TO BUFIO OUT S, STAT=',I5,'  LOOP=',I5)
      LLGDG=LGDG
  415 CONTINUE
C
C     OUTPUT RESIDUAL ARRAYS IF REQUIRED
C
      IF(NWFR) 420,425,420
  420 CONTINUE
      TIME=LO
      TT=TIME+0.01
      ITIM=TT
      ZIO(1,1;3*ION)=0.
      Z(1;ITOT)=ZS(1;ITOT)-ZT(1;ITOT)
      U(1;ITOT)=US(1;ITOT)-UT(1;ITOT)
      V(1;ITOT)=VS(1;ITOT)-VT(1;ITOT)
      ZIO(1,1;IINZ)=Q8VCMPRS(Z(1;ITOT),BITZ(1;ITOT);ZIO(1,1;IINZ))
      ZIO(1,2;IINZ)=Q8VCMPRS(U(1;ITOT),BITZ(1;ITOT);ZIO(1,2;IINZ))
      ZIO(1,3;IINZ)=O8VCMPRS(V(1;ITOT),BITZ(1;ITOT);ZIO(1,3;IINZ))
      ZIO(ION-4,3)=ITIM
      ZIO(ION-3,3)=IHS
      ZIO(ION-2,3)=IDAY
      ZIO(ION-1,3)=IMNTH
      ZIO(ION,3)=IYR
      CALL Q7BUFOUT(NWFR,ZIO,9,'SMALL')
  425 CONTINUE
C
C     PRINT OUTPUT
C
      IF(LLPR) 71,70,71
   70 CONTINUE
      TIME=LO
      IF(IPCL.EQ.0) GOTO 710
      Z(1;ITOT)=ZS(1;ITOT)-ZT(1;ITOT)
      U(1;ITOT)=US(1;ITOT)-UT(1;ITOT)
      V(1;ITOT)=VS(1;ITOT)-VT(1;ITOT)
      CALL PARFO(ITOT)
      IF(IPCP.EQ.0) GOTO 711
      Z(1;ITOT)=ZS(1;ITOT)
      U(1;ITOT)=US(1;ITOT)
      V(1;ITOT)=VS(1;ITOT)
      CALL PARFO(ITOT)
      Z(1;ITOT)=ZT(1;ITOT)
      U(1;ITOT)=UT(1;ITOT)
      V(1;ITOT)=VT(1;ITOT)
      CALL PARFO(ITOT)
  711 CONTINUE
  710 CONTINUE
      LLPR=LPR
   71 CONTINUE
C
C     CHECK RESIDUAL O/P COMPLETE
C
      IF(NWFR) 715,720,715
  715 CONTINUE
```

```
      CALL Q7WAIT(NWFR,ZIO,ISTAT,0,IRET)
      IF(ISTAT.NE.0) WRITE(NW,716) ISTAT,LOOP
  716 FORMAT(10X,' ABNORMAL END TO BUFIO OUT R, ISTAT=',I5,'  LOOP=',I5)
  720 CONTINUE
C
C     THE MAIN LOOP ENDS HERE
C
    9 CONTINUE
C
C     WRITE DATA TO FORECAST CONTROL FILE
C     SETS DEVICE NOS FOR INITIAL DATA NEXT FORECAST
      WRITE(NRWC) NWFT,NWFS,NRIT,NRIS
C
C     COMPUTE RESIDUAL TABLE
C
      ZETR(1,1;60*JET)=ZETS(1,1;60*JET)-ZETT(1,1;60*JET)
C
C     PRINT RESIDUAL TABLE
C
      CALL PELTA(JET,NPRTS,ITIT,ITIP,IHS,IDAY,IMNTH,IYR,IPCP)
C
C     STORE SELECTED PORTS
C
      IF(LPUN) 74,76,74
   74 CONTINUE
      WRITE(NWTT) IHS,IDAY,IMNTH,IYR,JET
      WRITE(NWTT) IZET
      WRITE(NWTT) ((ZETT(I,J),I=1,60),J=1,JET)
      WRITE(NWTT) ((ZETR(I,J),I=1,60),J=1,JET)
      END FILE NWTT
      WRITE(NW,137)
  137 FORMAT(1H0,35H  DATA FOR 60 PORTS WRITTEN TO DISC)
   76 CONTINUE
  990 CONTINUE
C
C     FINISH
C
      STOP
      END
      SUBROUTINE PARFO(ITOT)
      COMMON/COM1/Z(3072),U(3072),V(3072),ZZ(3072),UU(3072),VV(3072)
      COMMON/COM2/NW,N,NUM(60),NCC,NRR,NCON,ITSM
      COMMON/COM5/LLMI,LMI,ISWSC,TIME,IPCL,LO
      BIT BITZ,BITUX,BITVX,BITZD,BITUXD,BITVXD,BITIO
      DIMENSION BITZ(3072),BITUX(3072),BITVX(3072)
      COMMON/BUFIT/BITIO(32768)
      DESCRIPTOR BITZD,BITUXD,BITVXD
      DESCRIPTOR ZD,UD,VD,VVD
      EQUIVALENCE (BITZ(1),BITIO(1)),(BITUX(1),BITIO(9217)),
     C            (BITVX(1),BITIO(12289))
C
C     PREPARE ARRAYS FOR OUTPUT
C
      IA=IPCL-10
      IF(IA) 83,81,81
   81 CONTINUE
      ASSIGN BITZD,BITZ(1;ITOT)
      ASSIGN ZD,Z(1;ITOT)
      ASSIGN VVD,VV(1;ITOT)
      VVD=1001.0
      CALL Q8MASKV(X'00',,ZD,,VVD,BITZD,VVD)
      WRITE(NW,107) TIME
  107 FORMAT(1H1///////14H  ELEVATIONS  ,F14.8,7H  HOURS)
      CALL PRNTZ(VV)
      IF(IA) 86,86,83
   83 CONTINUE
      ASSIGN BITUXD,BITUX(1;ITOT)
      ASSIGN UD,U(1;ITOT)
      VVD=1001.0
      CALL Q8MASKV(X'00',,UD,,VVD,BITUXD,VVD)
      WRITE(NW,108) TIME
  108 FORMAT(1H1///////16H  U-COMPONENTS  ,F14.8,7H  HOURS)
      CALL PRNTZ(VV)
      ASSIGN BITVXD,BITVX(1;ITOT)
      ASSIGN VD,V(1;ITOT)
      VVD=1001.0
      CALL Q8MASKV(X'00',,VD,,VVD,BITVXD,VVD)
      WRITE(NW,109) TIME
  109 FORMAT(1H1///////16H  V-COMPONENTS  ,F14.8,7H  HOURS)
      CALL PRNTZ(VV)
   86 CONTINUE
      VV(1;ITOT)=0.0
      RETURN
      END
      SUBROUTINE PRNTZ(A)
      DIMENSION A(3072)
      COMMON/COM2/NW,N,NUM(60),NCC,NRR,NCON,ITSM
  110 FORMAT(1H //5X,16I7)
  111 FORMAT(2X,I2,3v,16F7.3)
  112 FORMAT(1H1//////5X,16I7)
      IS=1
      IND=16
      IDIF=IND-IS
      NCYC=(NCC/IND)+1
      NREM=NCC-(IND*(NCYC-1))
      IF(NREM.EQ.0) NCYC=NCYC-1
      WRITE(NW,110) (NUM(J),J=IS,IND)
      DO 99 ICYC=1,NCYC
      IF(ICYC.EQ.1) GO TO 98
      WRITE(NW,112) (NUM(J),J=IS,IND)
   98 CONTINUE
      DO 90 K=1,NRR
      I1=(K-1)*N+IS
      I2=I1+IDIF
      I3=NUM(K)
      WRITE(NW,111) I3,(A(I),I=I1,I2)
   90 CONTINUE
      IS=IS+IDIF
      IND=IS+IDIF
      IF(IND.LT.NCC) GO TO 99
      IND=NCC
      IDIF=NCC-IS
   99 CONTINUE
      RETURN
      END
      SUBROUTINE PELTA(JET,NPRTS,ITIT,ITIP,IHS,IDAY,IMNTH,IYR,IPCP)
      COMMON/COM2/NW,N,NUM(60),NCC,NRR,NCON,ITSM
      COMMON/COM3/ZETT(60,49),ZETS(60,49),ZET(60,49),IZET(60)
      CHARACTER*4 ITIT(20),ITIP(60)
      CHARACTER*1 ICHAR(3),IFLG(60,49)
      DATA ICHAR/' ','H','L'/
C
C     PRINT ELEVATION TABLES -  TO STWS SPECIFICATION
C
      DO 41 I=1,NPRTS
      DO 41 J=1,JET
      IFLG(I,J)=ICHAR(1)
   41 CONTINUE
      JETM=JET-1
      DO 40 J=2,JETM
      JO=J-1
      JN=J+1
      DO 40 I=1,NPRTS
      AOLD=ZETT(I,JO)
      A=ZETT(I,J)
      ANEW=ZETT(I,JN)
      IF(A.GT.AOLD.AND.A.GE.ANEW) IFLG(I,J)=ICHAR(2)
      IF(A.LT.AOLD.AND.A.LT.ANEW) IFLG(I,J)=ICHAR(3)
   40 CONTINUE
C     ONE TABLE PROVIDED     NO HINDCAST- ONE FORECAST
      ITAS=1
      IF(IPCP.EQ.0) ITAS=3
      DO 99 ITAB=ITAS,3
      IS=1
      IE=15
   43 CONTINUE
      JSTAR=1
      JSTOP=JET
      WRITE(NW,115)
  115 FORMAT(1H1)
      WRITE(NW,131) ITIT
  131 FORMAT(20A4)
      WRITE(NW,113) IHS,IDAY,IMNTH,IYR
  113 FORMAT(' DATA STARTS AT ',I3,' HRS GMT',I3,'/',I2,'/',I4)
      IF(ITAB.EQ.1) WRITE(NW,116)
  116 FORMAT(' TIDE + SURGE ELEVATIONS IN  M  '/)
      IF(ITAB.EQ.2) WRITE(NW,117)
  117 FORMAT(' TIDAL ELEVATIONS IN  M  '/)
      IF(ITAB.EQ.3) WRITE(NW,110)
  110 FORMAT(' RESIDUAL ELEVATIONS IN  M    '/)
      WRITE(NW,111) (IZET(I),I=IS,IE)
  111 FORMAT(1H0,'  TIME            POINT NUMBER'/'   GMT ',15I6)
      WRITE(NW,121) (ITIP(I),I=IS,IE)
  121 FORMAT(8X,15(1X,A4,1X))
      WRITE(NW,201)
  201 FORMAT(1H )
      DO 42 J=JSTAR,JSTOP
      ITIME=IHS+J-1
      IF(ITIME.GE.24) ITIME=ITIME-24
      ITIME=100*ITIME
      IF(J.EQ.7.OR.J.EQ.19) WRITE(NW,201)
      IF(ITAB.EQ.1) WRITE(NW,112) ITIME,((ZETS(I,J),IFLG(I,J)),I=IS,IE)
      IF(ITAB.EQ.2) WRITE(NW,112) ITIME,((ZETT(I,J),IFLG(I,J)),I=IS,IE)
      IF(ITAB.EQ.3) WRITE(NW,112) ITIME,((ZET(I,J),IFLG(I,J)),I=IS,IE)
  112 FORMAT(2X,I5,1X,15(F5.2,A1))
   42 CONTINUE
      IF(IE.GE.NPRTS) GOTO 44
      IS=IS+15
      IE=IE+15
      GOTO 43
   44 CONTINUE
   99 CONTINUE
      RETURN
      END
      SUBROUTINE SHPN(IIY,IVD,S,H,P,CN,P1)
      DIMENSION ENN(4)
      IY=IIY-1900
      JY = IY - 1
      IL = JY/4
      DL = IL + IVD - 1.0
      ENN(1) = 277.0247+129.38481*IY  +13.17639*DL
      ENN(2) = 280.1895-  0.23872*IY + 0.98565*DL
      ENN(3) = 334.3853+ 40.66249*IY  + 0.11140*DL
      ENN(4) = 259.1568- 19.32818*IY  - 0.05295*DL
      DO 20 J = 1, 4
      EN=ENN(J)
   10 IF(EN) 12,14,14
   12 EN=EN+360.
      GOTO 10
   14 IF(EN-360.) 18,16,16
   16 EN=EN-360.
      GOTO 14
   18 CONTINUE
      ENN(J)=EN
   20 CONTINUE
      S = ENN(1)
      H = ENN(2)
      P = ENN(3)
      CN = ENN(4)
      P1 = 282.2
      RETURN
      END
      SUBROUTINE DATEB(IDIN,IYIN,IDAY,IMN,IYEAR)
      DIMENSION ID(12),IM(12)
      DATA ID/31,28,31,30,31,30,31,31,30,31,30,31/
      IYR=IYIN
      IDY=IDIN
   99 CONTINUE
      IF(MOD(IYR,4).NE.0) ID(2)=28
      IF(MOD(IYR,4).EQ.0) ID(2)=29
      IF(MOD(IYR,100).EQ.0) ID(2)=28
      IF(MOD(IYR,400).EQ.0) ID(2)=29
      IM(1)=31
      DO 1 I=2,12
      IM(I)=IM(I-1)+ID(I)
    1 CONTINUE
      IF(IDY.LE.IM(12)) GOTO 98
      IDY=IDY-IM(12)
      IYR=IYR+1
      GOTO 99
   98 CONTINUE
      DO 2 I=1,12
      IF(IDY.LE.IM(I)) GOTO 3
    2 CONTINUE
    3 CONTINUE
      IF(I.GT.1) IDY=IDY-IM(I-1)
      IDAY=IDY
      IMN=I
      IYEAR=IYR
      RETURN
      END
```

```
 SUBROUTINE VDAY(IDAY,IMNTH,IY,IVDY)
IF(IMNTH.EQ.1) IVDY=IDAY
 IF(IMNTH.EQ.2) IVDY=IDAY+31
 IF(IMNTH.EQ.3) IVDY=IDAY+59
 IF(IMNTH.EQ.4) IVDY=IDAY+90
 IF(IMNTH.EQ.5) IVDY=IDAY+120
 IF(IMNTH.EQ.6) IVDY=IDAY+151
 IF(IMNTH.EQ.7) IVDY=IDAY+181
 IF(IMNTH.EQ.8) IVDY=IDAY+212
 IF(IMNTH.EQ.9) IVDY=IDAY+243
 IF(IMNTH.EQ.10) IVDY=IDAY+273
 IF(IMNTH.EQ.11) IVDY=IDAY+304
 IF(IMNTH.EQ.12) IVDY=IDAY+334
 IYR=IY
 IF(MOD(IYR,4).EQ.0.AND.IMNTH.GT.2) IVDY=IVDY+1
  IF(MOD(IYR,100).EQ.0.AND.IMNTH.GT.2) IVDY=IVDY-1
IF(MOD(IYR,400).EQ.0.AND.IMNTH.GT.2) IVDY=IVDY+1
 RETURN
 END
```

```
      SUBROUTINE GESMEQ
C***********************************************************************
C*
C*   SUBROUTINE GESMEQ - EQUATIONS OF CONTINUITY AND MOTION PLUS
C*                       BOUNDARY INPUTS
C*                       02/08/82 - CYBER
C*
C***********************************************************************
      DIMENSION H(3072),IZOB(150),IUOB(150),IVOB(150),COST(4),SIST(4),
     C   CORUA(3072),CORVA(3072),CSPUAI(3072),CSPVA(3072)
      DIMENSION TA(3072),TAU(3072),TAV(3072),TBU(3072),TBV(3072)
      DIMENSION BITH(3072),IZOBN(150),IZOBS(150),IZOBW(150),IZOBE(150)
      DIMENSION ZIN(150),TAP(3072),TAW(3072)
      DIMENSION BITZ(3072),BITU(3072),BITV(3072),BITUX(3072),BITVX(3072)
      DIMENSION BITZO(3072)
      DIMENSION NPIN(3,100),NPOT(3,100)
      BIT BITUX,BITVX,BITZO
      BIT BITIO
      BIT BITH,BITHD
      BIT BITZ,BITZD,BITU,BITUD,BITV,BITVD
      BIT BITZOD
      DESCRIPTOR BITHD,TBUD,TBVD
      DESCRIPTOR BITZD,BITUD,BITVD
      DESCRIPTOR ZD,ZZD,DZD,ZDD,UUD,VVD,ZTD,ZSD
      DESCRIPTOR UINTD,UINSD,VINTD,VINSD
      DESCRIPTOR BITZOD
      DESCRIPTOR TAPD,TAPDD,TAWD,TAD
      COMMON/BUFIT/BITIO(32768)
      COMMON/BUFRI/DATA(17408)
      COMMON/COM1/Z(3072),U(3072),V(3072),ZZ(3072),UU(3072),VV(3072)
      COMMON/COM2/NW,N,NUM(60),NCC,NRR,NCON,ITSM
      COMMON/COM4/DT,FR,DCRIT,ZCRIT,GC,RE,DTRX,DTRY,GDTRX,GDTRY,
     C            DTRO,FRDT,ITOV,ION,NTPH
      COMMON/COM5/LLHMI,LHMI,ISWSC,TIME,IPCL,LHO
      COMMON/COM6/ZB1(150),ZB2(150),UB1(150),UB2(150),VB1(150),VB2(150),
     C   Z1(150,4),Z2(150,4),U1(150,4),U2(150,4),V1(150,4),V2(150,4),
     C   SIG(4),ZINT(150),ZINS(150),P1(3072),P2(3072),F1(3072),F2(3072),
     C   G1(3072),G2(3072)
      EQUIVALENCE (BITZ(1),BITIO(1)),(BITU(1),BITIO(3073)),
     C            (BITV(1),BITIO(6145)),(BITUX(1),BITIO(9217)),
     C            (BITVX(1),BITIO(12289)),(BITZO(i),BITIO(15361))
      EQUIVALENCE (DATA(1),H(1)),(DATA(3073),CORUA(1)),
     C  (DATA(6145),CORVA(1)),(DATA(9217),CSPUAI(1)),
     C  (DATA(12289),CSPVA(1)),(DATA(15361),IZOB(1)),
     C  (DATA(15511),IUOB(1)),(DATA(15661),IVOB(1)),
     C  (DATA(15811),IZOBN(1)),(DATA(15961),IZOBS(1)),
     C  (DATA(16111),IZOBW(1)),(DATA(16261),IZOBE(1)),
     C  (NNRR,DATA(16500)),(NNCC,DATA(16501)),(ITOT,DATA(16502)),
     C  (IINZ,DATA(16503)),(IMEU,DATA(16504)),(IMEV,DATA(16505)),
     C  (IINUX,DATA(16506)),(IINVX,DATA(16507)),(IOBZ,DATA(16508)),
     C  (IOBU,DATA(16509)),(IOBV,DATA(16510)),(IOBZN,DATA(16511)),
     C  (IOBZS,DATA(16512)),(IOBZW,DATA(16513)),(IOBZE,DATA(16514)),
     C  (DX,DATA(16515)),(DY,DATA(16516))
      EQUIVALENCE (NTZ,DATA(16517)),(NTU,DATA(16518)),
     C   (NTV,DATA(16519)),(NPIN(1,1),DATA(16520)),
     C   (NPOT(1,1),DATA(16820))
      NCC=NNCC
      NRR=NNRR
      ITS=ITSM-1
      LMI=LHMI*NTPH
      LLMI=LLHMI*NTPH
      LO=(LHO-1)*NTPH
C
C     START TIMESTEPPING
C
      DO 1000 LOOP=1,NTPH
C
      LLMI=LLMI-1
      LO=LO+1
C
C     INITIALISE TEMP ARRAYS
C
      TA(1;ITOT)=0.0
      TAU(1;ITOT)=0.0
      TAV(1;ITOT)=0.0
      TBU(1;ITOT)=0.0
      TBV(1;ITOT)=0.0
C
C     CONTINUITY - EXPLICIT CODE
C
C
C     FLUX E-W
C
      ASSIGN TBUD,TBU(1;ITOT-1)
      ASSIGN TBVD,TBV(1;ITOT-N)
      TA(1;ITOT)=H(1;ITOT)+Z(1;ITOT)
      TAU(1;ITOT-1)=TA(1;ITOT-1)+TA(2;ITOT-1)
      TBU(1;ITOT-1)=0.5*TAU(1;ITOT-1)
      ASSIGN BITHD,BITH(1;ITOT-1)
      BITHD=BITHD.AND..NOT.BITHD
      BITHD=TBUD.LT.DCRIT
      CALL Q8VTOV(X'10',,DCRIT,,,BITHD,TBUD)
C     TBU STORES DEPTHS AT U POINTS FOR USE IN U EQUATION
C     CONTAINS H+Z AT U-POINTS
      TAU(N;ITOV+1)=U(N;ITOV+1)*TAU(N;ITOV+1)
C
C     FLUX N-S
C
      TAV(1;ITOT-N)=TA(1;ITOT-N)+TA(N+1;ITOT-N)
      TBV(1;ITOT-N)=0.5*TAV(1;ITOT-N)
      BITHD=BITHD.AND..NOT.BITHD
      ASSIGN BITHD,BITH(1;ITOT-N)
      BITHD=TBVD.LT.DCRIT
      CALL Q8VTOV(X'10',,DCRIT,,,BITHD,TBVD)
C     TBV STORES DEPTHS AT V POINTS FOR USE IN V EQUATION
C     CONTAINS H+Z AT V-POINTS
      TAV(1;ITOT-N)=V(1;ITOT-N)*TAV(1;ITOT-N)
      TAV(1;ITOT-N)=TAV(1;ITOT-N)*CSPVA(1;ITOT-N)
C
C     NET FLUX
C
      TAU(N+1;ITOV)=DTRX*(TAU(N+1;ITOV)-TAU(N;ITOV))
      TAV(N+1;ITOV)=DTRY*(TAV(1;ITOV)-TAV(N+1;ITOV))
      TAU(N+1;ITOV)=-0.5*(TAU(N+1;ITOV)+TAV(N+1;ITOV))
      TAU(N+1;ITOV)=TAU(N+1;ITOV)*CSPUAI(N+1;ITOV)
      ASSIGN ZD,Z(N+1;ITOV)
      ASSIGN ZZD,ZZ(N+1;ITOV)
      ASSIGN DZD,TAU(N+1;ITOV)
      ASSIGN BITZD,BITZ(N+1;ITOV)
      CALL Q8ADDNV(X'00',,ZD,,DZD,BITZD,ZZD)
C
C     TRANSFER ELEVATIONS FOR PATCH BOUNDARIES
C
      IF(NTZ.EQ.0) GOTO 11
      DO 10 I=1,NTZ
      JI=NPIN(1,I)
      JO=NPOT(1,I)
      ZZ(JO)=ZZ(JI)
   10 CONTINUE
   11 CONTINUE
C
C     CALCULATE CURRENTS ACROSS THE OPEN BOUNDARY
C
C     FIRST SET THE INTERPOLATION CONSTANTS FOR SURGES AND
C     COS AND SIN SIGMA T FOR TIDES
C
      A=FLOAT(LLMI)/FLOAT(LMI)
      B=1.0-A
      IF(ITS) 230,230,232
  230 CONTINUE
      TIME=LO*DT/3600.0
      ARG=3.14159265*TIME/180.0
      DO 231 ICON=1,NCON
      SARG=ARG*SIG(ICON)
      COST(ICON)=COS(SARG)
      SIST(ICON)=SIN(SARG)
  231 CONTINUE
  232 CONTINUE
C
C     NOW CALCULATE THE RESPONSE PART OF THE CURRENT
C
      ASSIGN ZTD,ZINT(1;IOBZ)
      ASSIGN ZSD,ZINS(1;IOBZ)
      ZTD=0
      ZSD=0
      IF(ITS.GT.0) GOTO 210
      DO 201 ICON=1,NCON
      ZTD=ZTD+Z1(1,ICON;IOBZ)*COST(ICON)+Z2(1,ICON;IOBZ)*SIST(ICON)
  201 CONTINUE
  210 CONTINUE
      IF(ITS.LT.0) GOTO 211
      ZSD=A*ZB1(1;IOBZ)+B*ZB2(1;IOBZ)
  211 CONTINUE
      ZIN(1;IOBZ)=ZTD+ZSD
C
C     NORTHERN OPEN BOUNDARY POINT
C
      IF(IOBZN.EQ.0) GOTO 206
      DO 202 J=1,IOBZN
      II=IZOBN(J)
      I=IZOB(II)
      D=ZZ(I)
      IMN=I-N
      H1=0.5*(H(I)+H(IMN)+D)
      C=SQRT(GC*H1)
  202 VV(IMN)=C*(D-ZIN(II))/H1
  206 CONTINUE
C
C     SOUTHERN OPEN BOUNDARY POINT
C
      IF(IOBZS.EQ.0) GOTO 207
      DO 203 J=1,IOBZS
      II=IZOBS(J)
      I=IZOB(II)
      D=ZZ(I)
      IPN=I+N
      H1=0.5*(H(I)+H(IPN)+D)
      C=SQRT(GC*H1)
  203 VV(I)=-C*(D-ZIN(II))/H1
  207 CONTINUE
C
C     WESTERN OPEN BOUNDARY POINT
C
      IF(IOBZW.EQ.0) GOTO 208
      DO 204 J=1,IOBZW
      II=IZOBW(J)
      I=IZOB(II)
      D=ZZ(I)
      H1=0.5*(H(I)+H(I-1)+D)
      C=SQRT(GC*H1)
  204 UU(I-1)=-C*(D-ZIN(II))/H1
  208 CONTINUE
C
C     EASTERN OPEN BOUNDARY POINT
C
      IF(IOBZE.EQ.0) GOTO 209
      DO 205 J=1,IOBZE
      II=IZOBE(J)
      I=IZOB(II)
      D=ZZ(I)
      H1=0.5*(H(I)+H(I+1)+D)
      C=SQRT(GC*H1)
  205 UU(I)=C*(D-ZIN(II))/H1
  209 CONTINUE
C
C     ADD THE TIDE AND SURGE INPUT CURRENT TO OBTAIN THE TOTAL
C
      ASSIGN UINTD,ZINT(1;IOBU)
      ASSIGN UINSD,ZINS(1;IOBU)
      UINTD=0
      UINSD=0
      IF(ITS.GT.0) GOTO 244
      DO 241 ICON=1,NCON
  241 UINTD=UINTD+U1(1,ICON;IOBU)*COST(ICON)+U2(1,ICON;IOBU)*SIST(ICON)
  244 CONTINUE
      IF(ITS.LT.0.OR.ISWSC.EQ.0) GOTO 247
      UINSD=A*UB1(1;IOBU)+B*UB2(1;IOBU)
  247 CONTINUE
      DO 245 J=1,IOBU
      I=IUOB(J)
  245 UU(I)=UU(I)+ZINT(J)+ZINS(J)
      ASSIGN VINTD,ZINT(1;IOBV)
      ASSIGN VINSD,ZINS(1;IOBV)
      VINTD=0
      VINSD=0
      IF(ITS.GT.0) GOTO 249
      DO 243 ICON=1,NCON
  243 VINTD=VINTD+V1(1,ICON;IOBV)*COST(ICON)+V2(1,ICON;IOBV)*SIST(ICON)
  249 CONTINUE
      IF(ITS.LT.0.OR.ISWSC.EQ.0) GOTO 251
      VINSD=A*VB1(1;IOBV)+B*VB2(1;IOBV)
  251 CONTINUE
      DO 248 J=1,IOBV
      I=IVOB(J)
  248 VV(I)=VV(I)+ZINT(J)+ZINS(J)
C
C     TRANSFER   CURRENTS FOR PATCH BOUNDARIES
C
      IF(NTU.EQ.0) GOTO 13
      DO 12 I=1,NTU
      JI=NPIN(2,I)
      JO=NPOT(2,I)
      UU(JO)=UU(JI)
   12 CONTINUE
   13 CONTINUE
      IF(NTV.EQ.0) GOTO 18
      DO 19 I=1,NTV
      JI=NPIN(3,I)
      JO=NPOT(3,I)
      VV(JO)=VV(JI)
   19 CONTINUE
   18 CONTINUE
C
C     U - EQUATION EXPLICIT CODE
C
C
C     WINDSTRESS AND AIR PRESSURE TERMS
C
      TAP(1;ITOT)=0.
      TAW(1;ITOT)=0.
      IF(ITSM.EQ.0) GOTO 260
      TAP(N+1;ITOV-1)=A*(P1(N+2;ITOV-1)-P1(N+1;ITOV-1))+
     C                B*(P2(N+2;ITOV-1)-P2(N+1;ITOV-1))
      TAW(N+1;ITOV-1)=A*F1(N+1;ITOV-1)+B*F2(N+1;ITOV-1)
      TAW(N+1;ITOV-1)=TAW(N+1;ITOV-1)*DTRO/TBU(N+1;ITOV-1)
  260 CONTINUE
C
C     PRESSURE GRAD
C
      TA(1;ITOT)=0.0
      TA(N+1;ITOV-1)=ZZ(N+2;ITOV-1)-ZZ(N+1;ITOV-1)
      TA(N+1;ITOV-1)=TA(N+1;ITOV-1)-TAP(N+1;ITOV-1)
      TA(N+1;ITOV-1)=GDTRX*(TA(N+1;ITOV-1)*CSPUAI(N+1;ITOV-1))
C
C     SUM SO FAR
C
      TA(N+1;ITOV-1)=U(N+1;ITOV-1)-TA(N+1;ITOV-1)+TAW(N+1;ITOV-1)
C
C     V-AVERAGE
C
```

```
      TAP(1;ITOT)=0.0
      TAU(1;ITOT)=0.0
      TAP(1;ITOT-1)=V(1;ITOT-1)+V(2;ITOT-1)
      TAU(N+1;ITOV-1)=0.25*(TAP(1;ITOV-1)+TAP(N+1;ITOV-1))
C
C     FRICTION TERM
C
      TAW(1;ITOT)=0.0
      TAV(1;ITOT)=0.0
      TAW(1;ITOT)=U(1;ITOT)*U(1;ITOT)
      TAV(N+1;ITOV-1)=TAW(N+1;ITOV-1)+TAU(N+1;ITOV-1)*TAU(N+1;ITOV-1)
      TAV(N+1;ITOV-1)=VSQRT(TAV(N+1;ITOV-1);TAV(N+1;ITOV-1))
      TAV(N+1;ITOV-1)=1.0+FRDT*(TAV(N+1;ITOV-1)/TBU(N+1;ITOV-1))
C
C     CORIOLIS TERM
C
      TAU(N+1;ITOV-1)=DT*TAU(N+1;ITOV-1)*CORUA(N+1;ITOV-1)
      TA(N+1;ITOV-1)=TA(N+1;ITOV-1)+TAU(N+1;ITOV-1)
C
C     ADVECTION FORWARD IN TIME
C
      TAU(1;ITOT)=0.0
      TAU(1;ITOT-N)=DTRY*(U(1;ITOT-N)-U(N+1;ITOT-N))
      TAU(1;ITOT-N)=TAU(1;ITOT-N)*TAP(1;ITOT-N)
      TAU(N+1;ITOV-1)=TAU(1;ITOV-1)+TAU(N+1;ITOV-1)
      TAP(N+1;ITOV-1)=DTRX*(TAW(N+2;ITOV-1)-TAW(N;ITOV-1))*
     C    CSPUAI(N+1;ITOV-1)
      TAU(N+1;ITOV-1)=0.25*(TAU(N+1;ITOV-1)+TAP(N+1;ITOV-1))
      ASSIGN BITUD,BITU(N+1;ITOV-1)
      ASSIGN BITZOD,BITZO(N+1;ITOV-1)
      WHERE(BITUD.AND..NOT.(BITUD.AND.BITZOD)) TA(N+1;ITOV-1)=
     C TA(N+1;ITOV-1)+TAU(N+1;ITOV-1)
      TA(N+1;ITOV-1)=TA(N+1;ITOV-1)/TAV(N+1;ITOV-1)
C
C     MASK OUT UNWANTED ELEMENTS
C
      ASSIGN ZDD,TA(N+1;ITOV-1)
      ASSIGN UUD,UU(N+1;ITOV-1)
      CALL Q8MASKV(X'00',,ZDD,,UUD,BITUD,UUD)
C
C     DRYING CONDITION
C
      ASSIGN ZDD,TAU(N+1;ITOV-1)
      ASSIGN TAD,TA(N+1;ITOV-1)
      ASSIGN TAPD,TAP(N+1;ITOV-1)
      ASSIGN TAPDD,TAP(N+2;ITOV-1)
      ASSIGN TAWD,TAW(N+1;ITOV-1)
      TAP(1;ITOT)=0.
      TA(1;ITOT)=0.
      TAW(1;ITOT)=0.
      TAPD=H(N+1;ITOV-1)+ZZ(N+1;ITOV-1)
      TAPDD=H(N+2;ITOV-1)+ZZ(N+2;ITOV-1)
      TAD=TAPD+TAPDD
      TAWD=ZZ(N+1;ITOV-1)-ZZ(N+2;ITOV-1)
      TAU(1;ITOT)=0.
      WHERE (((TAPD.GT.0).AND.(TAPDD.GT.0)).OR.((TAPD.GT.0).AND.(TAPDD.
     1LE.0).AND.(TAD.GT.0).AND.((TAWD-ZCRIT).GT.0)).OR.((TAPD.LE.0).AND.
     2(TAPDD.GT.0).AND.(TAD.GT.0).AND.((TAWD+ZCRIT).LT.0))) ZDD=UUD
```

```
      CALL Q8MASKV(X'00',,ZDD,,UUD,BITVD,UUD)
C
C     TRANSFER U-CURRENTS FOR PATCH BOUNDARIES
C
      IF(NTU.EQ.0) GOTO 15
      DO 14 I=1,NTU
      JI=NPIN(2,I)
      JO=NPOT(2,I)
      UU(JO)=UU(JI)
   14 CONTINUE
   15 CONTINUE
C
C     V - EQUATION EXPLICIT CODE
C
C
C     WINDSTRESS AND AIR PRESSURE TERMS
C
      TAP(1;ITOT)=0.
      TAW(1;ITOT)=0.
      IF(ITSM.EQ.0) GOTO 265
      TAP(N+1;ITOV)=A*(P1(N+1;ITOV)-P1(2*N+1;ITOV))+
     C               B*(P2(N+1;ITOV)-P2(2*N+1;ITOV))
      TAW(N+1;ITOV)=A*G1(N+1;ITOV)+B*G2(N+1;ITOV)
      TAW(N+1;ITOV)=TAW(N+1;ITOV)*DTRO/TBV(N+1;ITOV)
  265 CONTINUE
C
C     PRESSURE GRAD
C
      TA(1;ITOT)=0.0
      TA(N+1;ITOV)=ZZ(N+1;ITOV)-ZZ(2*N+1;ITOV)
      TA(N+1;ITOV)=GDTRY*(TA(N+1;ITOV)-TAP(N+1;ITOV))
C
C     SUM SO FAR
C
      TA(N+1;ITOV)=V(N+1;ITOV)-TA(N+1;ITOV)+TAW(N+1;ITOV)
C
C     U AVERAGES
C
      TAP(1;ITOT)=0.0
      TAU(1;ITOT)=0.0
      TAP(1;ITOT-N)=U(1;ITOT-N)+U(N+1;ITOT-N)
      TAU(N+1;ITOV)=0.25*(TAP(N;ITOV)+TAP(N+1;ITOV))
C
C     FRICTION TERM
C
      TAW(1;ITOT)=0.0
      TBU(1;ITOT)=0.0
      TAW(1;ITOT)=V(1;ITOT)*V(1;ITOT)
      TBU(N+1;ITOV)=TAU(N+1;ITOV)*TAU(N+1;ITOV)
      TBU(N+1;ITOV)=TBU(N+1;ITOV)+TAW(N+1;ITOV)
      TBU(N+1;ITOV)=VSORT(TBU(N+1;ITOV);TBU(N+1;ITOV))
      TBU(N+1;ITOV)=1.0+FRDT*(TBU(N+1;ITOV)/TBV(N+1;ITOV))
C
C     CORIOLIS
C
      TAV(1;ITOT)=0.0
      TAV(N+1;ITOT-N)=UU(N;ITOT-N)+UU(N+1;ITOT-N)
      TAV(N+1;ITOV)=0.25*(TAV(N+1;ITOV)+TAV(2*N+1;ITOV))
      TAV(N+1;ITOV)=DT*TAV(N+1;ITOV)*CORVA(N+1;ITOV)
      TA(N+1;ITOV)=TA(N+1;ITOV)-TAV(N+1;ITOV)
C
C     ADVECTION FORWARD IN TIME
C
      TAU(1;ITOT)=0.0
      TAU(1;ITOT-1)=DTRX*(V(2;ITOT-1)-V(1;ITOT-1))
      TAU(1;ITOT-N)=TAP(1;ITOT-N)*TAU(1;ITOT-N)/CSPVA(1;ITOT-N)
      TAU(N+1;ITOV)=TAU(N;ITOV)+TAU(N+1;ITOV)
      TAP(N+1;ITOV)=DTRY*(TAW(1;ITOV)-TAW(2*N+1;ITOV))
      TAU(N+1;ITOV)=0.25*(TAU(N+1;ITOV)+TAP(N+1;ITOV))
      ASSIGN BITVD,BITV(N+1;ITOV)
      ASSIGN BITZOD,BITZO(N+1;ITOV)
      WHERE(BITVD.AND..NOT.(BITVD.AND.BITZOD)) TA(N+1;ITOV)=
     C  TA(N+1;ITOV)-TAU(N+1;ITOV)
      TA(N+1;ITOV)=TA(N+1;ITOV)/TBU(N+1;ITOV)
C
C     MASK OUT UNWANTED ELEMENTS
C
      ASSIGN ZDD,TA(N+1;ITOV)
      ASSIGN VVD,VV(N+1;ITOV)
      CALL Q8MASKV(X'00',,ZDD,,VVD,BITVD,VVD)
C
C     DRYING CONDITION
C
      ASSIGN ZDD,TAU(N+1;ITOV)
      ASSIGN TAD,TA(N+1;ITOV)
      ASSIGN TAPD,TAP(N+1;ITOV)
      ASSIGN TAPDD,TAP(2*N+1;ITOV)
      ASSIGN TAWD,TAW(N+1;ITOV)
      TAP(1;ITOT)=0.
      TA(1;ITOT)=0.
      TAW(1;ITOT)=0.
      TAPD=H(N+1;ITOV)+ZZ(N+1;ITOV)
      TAPDD=H(2*N+1;ITOV)+ZZ(2*N+1;ITOV)
      TAD=TAPD+TAPDD
      TAWD=ZZ(N+1;ITOV)-ZZ(2*N+1;ITOV)
      TAU(1;ITOT)=0.
      WHERE (((TAPD.GT.0).AND.(TAPDD.GT.0)).OR.((TAPD.GT.0).AND.(TAPDD.
     1LE.0).AND.(TAD.GT.0).AND.((TAWD-ZCRIT).GT.0)).OR.((TAPD.LE.0).AND.
     2(TAPDD.GT.0).AND.(TAD.GT.0).AND.((TAWD+ZCRIT).LT.0))) ZDD=VVD
      CALL Q8MASKV(X'00',,ZDD,,VVD,BITVD,VVD)
C
C     TRANSFER V-CURRENTS FOR PATCH BOUNDARIES
C
      IF(NTV.EQ.0) GOTO 17
      DO 16 I=1,NTV
      JI=NPIN(3,I)
      JO=NPOT(3,I)
      VV(JO)=VV(JI)
   16 CONTINUE
   17 CONTINUE
C
C     NEW Z U V MOVED TO OLD ARRAYS
C
      Z(1;ITOT)=ZZ(1;ITOT)
      U(1;ITOT)=UU(1;ITOT)
      V(1;ITOT)=VV(1;ITOT)
```

```
C
C     END TIMESTEPPING
C
 1000 CONTINUE
C
C
      RETURN
      END
      SUBROUTINE METPROC(ITOT,A0,A1,ROA,ROW,GE,PMEAN,ISET,IFREQ)
      DIMENSION W(17408),TAE(3072),TAN(3072),PO(3072),UO(3072),VO(3072)
      COMMON/BUFWI/WH(8704)
      COMMON/BUFPH/PH(4352)
      DIMENSION BITW(8704),BITP(8704)
      DIMENSION IIMU(3072),IIMV(3072),IIMZ(3072),
     C         AUX(3072),AVX(3072),AZX(3072),BUX(3072),BVX(3072),
     C         BZX(3072)
      COMMON/BUMIT/BITIM(32768)
      EQUIVALENCE (BITW(1),BITIM(1)),(BITP(1),BITIM(8705))
      COMMON/BUMRI/DAM(28160)
      EQUIVALENCE (IIMU(1),DAM(1)),(IIMV(1),DAM(3073)),
     C            (IIMZ(1),DAM(6145)),(AUX(1),DAM(9217)),
     C            (AVX(1),DAM(12289)),(AZX(1),DAM(15361)),
     C            (BUX(1),DAM(18433)),(BVX(1),DAM(21505)),
     C            (BZX(1),DAM(24577)),(IWO,DAM(27649)),
     C            (NWEL,DAM(27650)),(NCCMS,DAM(27652)),
     C            (NRRMS,DAM(27653)),(IPO,DAM(27654)),
     C            (NPEL,DAM(27655)),(NCCPS,DAM(27657))
     C            (NRRPS,DAM(27658))
      BIT BITW,BITWD,BITIM
      BIT BITP,BITPD
      DESCRIPTOR BITWD,WHD,WD,UOD,VOD,WDU,WDV,TAED,TAND,WPD,PHD,POD
      DESCRIPTOR ISEAD,IIMUD,IIMVD,BITPD
      DESCRIPTOR AUXD,BUXD,AVXD,BVXD
      COMMON/COM2/NW,N,NUM(60),NCC,NRR,NCON,ITSM
      COMMON/COM6/ZB1(150),ZB2(150),UB1(150),UB2(150),VB1(150),VB2(150),
     C   Z1(150,4),Z2(150,4),U1(150,4),U2(150,4),V1(150,4),V2(150,4),
     C   SIG(4),ZINT(150),ZINS(150),P1(3072),P2(3072),F1(3072),F2(3072),
     C   G1(3072),G2(3072)
C
C     CONVERT WIND DATA FROM HALF TO FULL PRECISION
C
      ASSIGN WHD,WH(1;17408)
      ASSIGN WD,W(1;17408)
      CALL Q8EXTV(,,WHD,,,,WD)
C
C     EXTRACT WIND DATA AT POINTS REQUIRED
C
      ASSIGN UOD,UO(1;IWO)
      ASSIGN VOD,VO(1;IWO)
      ASSIGN WDU,W(1;NWEL)
      ASSIGN WDV,W(NWEL+1;NWEL)
      ASSIGN BITWD,BITW(1;NWEL)
      UOD=Q8VCMPRS(WDU,BITWD;UOD)
      VOD=Q8VCMPRS(WDV,BITWD;VOD)
C
C     PRINT WINDS
C
      ISWOP=ISET/IFREQ
      ISWOP=ISWOP*IFREQ
      IF(IFREQ.LT.0) ISWOP=-1
      IF(ISWOP.NE.ISET) GOTO 998
      CALL PRIW(NW,1,NCCMS,1,NRRMS,UO,VO,2)
  998 CONTINUE
C
C     COMPUTE WIND STRESS COMPONENTS
C
```

```
      ASSIGN TAED,TAE(1;IWO)
      ASSIGN TAND,TAN(1;IWO)
      TAED=UOD*UOD
      TAND=VOD*VOD
      TAED=TAED+TAND
      TAND=VSQRT(TAED;TAND)
C
C     CONTAINS WIND SPEED IN M/S
C
      TAED=(A0+A1*TAND)*0.001*ROA
C
C     WIND STRESS COEFFICIENT
C
      UOD=UOD*TAED
      VOD=VOD*TAED
      UOD=UOD*TAND
      VOD=VOD*TAND
C
C     UO AND VO NOW CONTAIN E AND N COMPONENTS OF WIND STRESS
C
      IF(ISWOP.NE.ISET) GOTO 997
      CALL PRIW(NW,1,NCCMS,1,NRRMS,UO,VO,2)
  997 CONTINUE
C
C     CONVERT PRESSURE DATA FROM HALF TO FULL PRECISION
C
      ASSIGN WPD,W(1;NPEL)
      ASSIGN PHD,PH(1;NPEL)
      CALL Q8EXTV(,,PHD,,,,WPD)
C
C     EXTRACT PRESSURE DATA AT POINTS REQUIRED
C
      ASSIGN POD,PO(1;IPO)
      ASSIGN BITPD,BITP(1;NPEL)
      POD=Q8VCMPRS(WPD,BITPD;POD)
C
C     PRINT PRESSURE DEVIATIONS FROM 1000MB
C
      IF(ISWOP.NE.ISET) GOTO 995
      POD=POD-1000.
      CALL PRIW(NW,1,NCCPS,1,NRRPS,PO,PO,1)
      POD=POD+1000.
  995 CONTINUE
      POD=PMEAN-POD
      POD=100.*POD/(ROW*GE)
C
C     PO NOW CONTAINS EQUIVALENT HYDROSTATIC SEA SURFACE
C     ELEVATION IN METRES
C
      IF(ISWOP.NE.ISET) GOTO 994
      CALL PRIW(NW,1,NCCPS,1,NRRPS,PO,PO,1)
  994 CONTINUE
C
C     LINEAR INTERPOLATION TO SEA MODEL POINTS
C
      CALL LIMP(IIMU,AUX,BUX,ITOT,NCCMS,UO)
C
C     UO NOW CONTAINS STRESSES AT U POINTS IN THE SEA MODEL
```

```
C
      CALL LIMP(IIMV,AVX,BVX,ITOT,NCCMS,VO)
C
C     VO NOW CONTAINS STRESSES AT V POINTS IN THE SEA MODEL
C
      CALL LIMP(IIMZ,AZX,BZX,ITOT,NCCPS,PO)
C
C     PO NOW CONTAINS HYDROSTATIC ELEVATIONS AT Z POINTS IN SEA MODEL
C
      IF(ISWOP.NE.ISET) GOTO 996
      CALL PRIW(NW,1,NCC,1,NRR,UO,VO,2)
      CALL PRIW(NW,1,NCC,1,NRR,PO,PO,1)
  996 CONTINUE
C
C     TRANSFER RESULTS TO STRESS AND PRESSURE ARRAYS AND RETURN
C
      P2(1;ITOT)=PO(1;ITOT)
      F2(1;ITOT)=UO(1;ITOT)
      G2(1;ITOT)=VO(1;ITOT)
      RETURN
      END
      SUBROUTINE PRIW(NW,IS,IE,JS,JE,UO,VO,NCOMP)
      DIMENSION UO(3072),VO(3072)
      JSTOP=JE-JS+1
      IEO=IE-IS+1
      NP=(IEO/20)
      IF(IEO.NE.NP*20) NP=NP+1
      ICOMP=1
      WRITE(NW,100) ICOMP
  100 FORMAT(16I5)
      DO 12 IP=1,NP
      ISC=(IP-1)*20+1
      IEC=IP*20
      IF(IEC.GT.IEO) IEC=IEO
      DO 10 JR=1,JSTOP
      I1=(JR-1)*IEO+ISC
      I2=(JR-1)*IEO+IEC
      WRITE(NW,101) (UO(I),I=I1,I2)
  101 FORMAT(2X,20F6.2)
   10 CONTINUE
      WRITE(NW,102)
  102 FORMAT(1H0)
   12 CONTINUE
      IF(NCOMP.EQ.1) GOTO 99
      ICOMP=2
      WRITE(NW,100) ICOMP
      DO 13 IP=1,NP
      ISC=(IP-1)*20+1
      IEC=IP*20
      IF(IEC.GT.IEO) IEC=IEO
      DO 11 JR=1,JSTOP
      I1=(JR-1)*IEO+ISC
      I2=(JR-1)*IEO+IEC
      WRITE(NW,101) (VO(I),I=I1,I2)
   11 CONTINUE
      WRITE(NW,102)
   13 CONTINUE
   99 CONTINUE
      RETURN
      END
      SUBROUTINE LIMP(IIMU,AUX,BUX,ISO,NCCMS,UO)
C
C     LINEAR INTERPOLATION FROM MET TO SEA MODEL
C
      DIMENSION IIMU(3072),AUX(3072),BUX(3072),UO(3072),IIM(3072),
     C        F1(3072),F2(3072),F3(3072),F4(3072)
      DESCRIPTOR F1D,F2D,F3D,F4D,IIMUD,IIMD
      DESCRIPTOR UOD,AUXD,BUXD
C
C     FILL VECTORS FOR INTERPOLATION
C
      ASSIGN IIMUD,IIMU(1;ISO)
      ASSIGN IIMD,IIM(1;ISO)
      ASSIGN F1D,F1(1;ISO)
      ASSIGN F2D,F2(1;ISO)
      ASSIGN F3D,F3(1;ISO)
      ASSIGN F4D,F4(1;ISO)
      ASSIGN UOD,UO(1;ISO)
      ASSIGN AUXD,AUX(1;ISO)
      ASSIGN BUXD,BUX(1;ISO)
      F1D=0
      F2D=0
      F3D=0
      F4D=0
      CALL Q8VXTOV(X'00',,IIMUD,,UOD,,F1D)
C
C     F1 CONTAINS U STRESS AT I POINTS
C
      IIMD=IIMUD+1
      CALL Q8VXTOV(X'00',,IIMD,,UOD,,F2D)
C
C     F2 CONTAINS U STRESS AT I+1 POINTS
C
      IIMD=IIMD+NCCMS
      CALL Q8VXTOV(X'00',,IIMD,,UOD,,F4D)
C
C     F4 CONTAINS U STRESS AT I+NM+1 POINTS
C
      IIMD=IIMD-1
      CALL Q8VXTOV(X'00',,IIMD,,UOD,,F3D)
C
C     F3 CONTAINS U STRESS AT I+NM POINTS
C
C     INTERPOLATE
C
      UOD=0
      F1D=F1D*(1.0-AUXD)
      F2D=F2D*AUXD+F1D
      F3D=F3D*(1.0-AUXD)
      F4D=F4D*AUXD+F3D
      UOD=(1.0-BUXD)*F2D
      UOD=UOD+BUXD*F4D
      RETURN
      END
      SUBROUTINE STARTI(NW,JTIM,IHLAP)
      DIMENSION JTIM(5,4),IHLAP(4)
C
C     COMPUTE ELAPSED TIME HRS SINCE 0000GMT 1/1/1982
C     FOR WIND, PRESSURE, TIDE AND TIDE + SURGE DATA
C
      WRITE(NW,101)
  101 FORMAT(/////,' DATA START TIMES FOR WIND, PRESSURE, TIDE AND TIDE
     &+ SURGE ')
      DO 1 J=1,4
      ICNT=0
      IYR=JTIM(5,J)
      IMNTH=JTIM(4,J)
      IDAY=JTIM(3,J)
      IHS=JTIM(2,J)
      ITIME=JTIM(1,J)
      IF(IYR.LE.1982) GOTO 10
      IYEX=IYR-1
      DO 2 IY=1982,IYEX
      CALL VDAY(31,12,IY,IIDY)
      ICNT=ICNT+IIDY
    2 CONTINUE
   10 CONTINUE
      CALL VDAY(IDAY,IMNTH,IYR,IIDY)
      ICNT=ICNT+IIDY-1
      ICNT=24*ICNT+IHS+ITIME
      IHLAP(J)=ICNT
C     NUMBER OF HOURS ELAPSED SINCE 0000GMT 1/1/1982
      WRITE(NW,100) J,(JTIM(I,J),I=1,5),IHLAP(J)
  100 FORMAT(10X,6I5,I10)
    1 CONTINUE
C
C     NOW RELATE ALL TIMES TO START OF WIND DATA
C
      I1=IHLAP(1)
      DO 3 J=1,4
    3 IHLAP(J)=IHLAP(J)-I1
      WRITE(NW,102) (IHLAP(J),J=1,4)
  102 FORMAT(10X,8I10)
      I1=IHLAP(1)
      I2=IHLAP(2)
      I3=IHLAP(3)
      I4=IHLAP(4)
      IF(I1.NE.I2) WRITE(NW,310) I1,I2
  310 FORMAT(10X,' WARNING **********  START TIMES OF WIND AND PRESSURE
     &DATA DIFFER  WIND= ',I5,' PRESSURE= ',I5,'  TIME OF WIND USED ')
      IF(I3.NE.I4) WRITE(NW,311) I3,I4
  311 FORMAT(10X,' WARNING **********  START TIMES OF TIDE AND SURGE DA
     &TA DIFFER  TIDE= ',I5,' SURGE= ',I5,'  TIME OF SURGE USED ')
      RETURN
      END
```