KONINKLIJK NEDERLANDS
METEOROLOGISCH INSTITUUT

De Bilt

WETENSCHAPPELIJK RAPPORT
W.R. 75-13

G.J. Prangsma

On the automatic handling of data obtained from recording current meters

De Bilt, 1975.

Publikatienummer: K.N.M.I. W.R. 75-13 (00)

U.D.C.: 551.46.085 :
551.46.071 :
681.3:518

Samenvatting.

Sinds in 1968 zelfregistrerende stroommeters door het KNMI in gebruik werden genomen, is aan de geautomatiseerde behandeling van de verzamelde gegevens grote aandacht geschonken.

In een rapport (V-241, 1972) geven de Crook en van der Veen een beschrijving van programma's van de eerste verwerkingsfase. Door het snel toenemende gebruik van de registrerende stroommeters bleek een verdergaande automatisering nodig, terwijl tevens een hoeveelheid ervaring beschikbaar kwam, die aanpassing van de bestaande computer programma's vergde. In 1972 werd daarom besloten een nieuw programma-pakket te ontwikkelen met de volgende doelstellingen:

- vergaande geautomatiseerde eliminatie van fouten. Deze fouten kunnen ondermeer hun oorsprong vinden in:
  a. de sensoren in de stroommeter
  b. de digitalisering in de stroommeter
  c. de magneetband registratie
  d. storingen in de werking van de klok in de stroommeter
  e. de vertaling van magneetband naar ponsband.
- de in ICES-verband gewenste data-produkten berekenen en presenteren in de gevraagde vorm.
- toekomstige uitbreidingen van of veranderingen in het pakket moeten op eenvoudige wijze in te bouwen zijn.
- de rekentijd van de bestaande programma's mag niet overschreden worden, doch liefst bekort.

Met de programma's zoals die in dit rapport beschreven staan, is aan bovengenoemde eisen voldaan.

Dit is ondermeer bereikt door:

- een konsekwente toepassing van de bufferingstechniek beschreven in hoofdstuk 2, welke is geschreven in een hogere programmeertaal (Algol). Hiermee is het gelukt om op die plaatsen, waar niet regeldrukker en/of bandponser de snelheid bepalende apparaten zijn, de rekensnelheid van de EL-X8 installatie volledig te benutten.
- de in hoofdstuk 3 (paragraaf 2) beschreven analyse methode voor de invoergegevens, die gebruik maakt van de bekende struktuur van deze invoerstroom.
- test-kriteria te ontwikkelen voor de verschillende meetkanalen, die ongevoelig zijn voor het karakter van de getijstroom ter plaatse van de meting (paragrafen 3.3 t/m 3.6).

- aanvullend op de testkriteria interpolatie-methoden voor gesignaleerde fouten (hoofdstuk 4).
- modulaire opbouw van de programma's.

Hiermee is bereikt, dat slechts een geringe visuele inspectie nodig is om een gegevensbestand te verkrijgen dat voor wat de betrouwbaarheid betreft als het best bereikbare kan worden beschouwd.

De data-produkten die berekend worden betreffende (hoofdstuk 5) uurlijks en getij-gemiddelde stromen en temperaturen.

De beschreven programma's zijn (nog) niet in staat om alle, mogelijke fouten te verbeteren, terwijl ook enkele wensen zijn blijven liggen (hoofdstuk 6). Wellicht dat de overgang op een nieuwe computer-installatie van het KNMI gelegenheid biedt hier nog aandacht aan te besteden.

CONTENTS.

References

Appendix A. Summary of the EL-X8 I/O functions used in the programs DATA-DECODER-MULTIPURPOSE, DATA-COMPUTATION and ACM.

Appendix B. Description of the initial and calibration data area on drum storage.

Appendix C. Input documents and conventions.

C-1. Forms for the DATA-DECODER-MULTIPURPOSE program.

C-2. Preparation of the 8-track translator tape.

C-3. Forms for the ACM program.

Appendix D. DATA-DECODER-MULTIPURPOSE.

Appendix E. DATA-COMPUTATION.

Appendix F. ACM (Actual Time Current Meters).

1. Introduction.

Since the introduction of automatically recording current meters in the observational programme of the department for oceanographic research of the Royal Netherlands Meteorological Institute (KNMI) in 1968, the processing of the data obtained with this instrumentation has been of continuing importance. On the one hand all information contained in the registrations should become available, whereas on the other hand limitations on manpower and computing facilities necessitated compromise. This compromise consisted of a computer programme which decoded the punched paper tape produced by a translation of the magnetic recorder tape of the Plessey and Aanderaa current meters, and then produced a listing of the decoded and computed values.

A description (in Dutch) of this programme is given in ref. (1).

Intensification of the observational programme, restrictions in available manpower and computer time and the necessity to produce averaged data for exchange among participants in the ICES Pilot Current Meter Network required re-considering of the software package.

Therefore in 1972 a reconstruction of the package was started with the following aims:

- detection and elimination of as much errors as reasonably possible by the computer.
- computation of the data products agreed upon for exchange under ICES.
- implementation of future additions or alterations must be readily possible.
- computer time the same as or less than the existing package.

The programmes described in this report have shown to meet the above aims by using the following ingredients:

a. buffering techniques on back up storage (written in a high level language). This technique allows the Institute's EL-X8-computer to run at full speed unless line printer or punched paper tape equipment is involved.

   A description is given in chapter 2.

b. error detection (as described in chapter 3) is done at a number of levels:

   1. the (known) structure of the input data stream is used as a tool to detect errors in the magnetic tape recording in the current meter and errors in the translation process of this magnetic tape to a punched paper tape (paragraph 3.2).

2. test criteria have been developed for the data channels (paragraph 3.3 - 3.5). These criteria are such as to be independent of the tidal characteristics at the mooring locations.

   The speed criterion also acts as a detector for possible errors in the performance of the current meter clock (paragraph 3.6).

c. Complementary to the test criteria are the interpolation procedures for data, which are marked as possible errors (chapter 4).

d. The data products computed are hourly and tidal mean values for speed components and temperature (chapter 5).

In chapter 6 suggestions are given for possible additional modules that could not be implemented on our present computer installation.

The programmes outlined in this report have been written in the ALGOL-60 language. Source listings are given in the Appendices D, E and F.

Presentation of the programmes does by no means imply that they are without errors. Some errors do occur, be it only once in our present experience. However, it is hoped that a re-analysis and re-write of the programmes for the new computer configuration of this Institute by the fall of 1975 can overcome some of the errors encountered.

2. Buffering techniques and layout of data on temporary (drum) storage.

2.1. Buffering techniques.

In all three programmes discussed in this report extensive use is made of a quasi-virtual memory technique whenever the large amount of current meter data is involved. This is achieved by using twin buffers which are alternatingly read (or filled) by the programme and transferred to (or from) the drum storage, for each file.

To this end each file is accessed via a subroutine of the structure outlined in fig. 2-2 for the case of reading from a file. Initialization of the process is achieved by the sequence shown

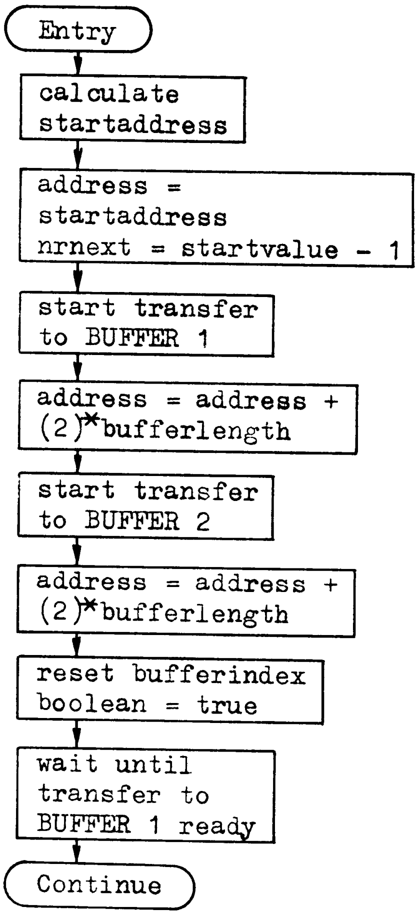

Entry
calculate startaddress
address = startaddress
nrnext = startvalue - 1
start transfer to BUFFER 1
address = address + (2)*bufferlength
start transfer to BUFFER 2
address = address + (2)*bufferlength
reset bufferindex
boolean = true
wait until transfer to BUFFER 1 ready
Continue

Fig. 2-1 Buffer initialization sequence for reading data from a file in fig. 2-1.

From the starting information the drum address of the first memory word is computed and used in the initialization of the transfer to BUFFER 1.

The address is then increased by the bufferlength to initialize the transfer to BUFFER 2. The address is then increased once more for subsequent use in the subroutine in fig. 2-2.

Further variables are set to the proper values to be used in the subroutine whereas the "boolean" is set to direct the subroutine to read first from BUFFER 1. The last step of the initialization sequence is to uphold further actions until BUFFER 1 has been filled.

The subroutine outlined in fig. 2-2 can then be used to read data from the appropriate buffer to,say, an array DATA which is used in the main programme. For certain purposes the use of a counter "nrnext" is needed which is incremented upon each call of the subroutine.

The "boolean" is tested to direct the execution to the buffer currently in use. After transscribing of the buffer to DATA the bufferindex is increased by the appropriate amount-depending on the length of DATA- and compared with the bufferlength.

If the buffer is exhausted the following actions are performed:

- a refill is initiated
- the bufferindex is reset
- the address is increased by the bufferlength for integer type variables or by 2*bufferlengths if real variables are involved.
- boolean is negated, i.e. set to point to the other buffer upon the next entry of the subroutine.
- further execution is delayed until filling of the buffer to be used next is completed.

This scheme has been developed for sequential work through the file. Clearly jumping backward (or forward) can be achieved be an execution of the initialization sequence with appropriate starting information.

This sort of "random access" should be minimized, however, to keep execution times as low as possible.

By choosing the bufferlength the time needed for the transfers to/from drum storage can be tuned to the execution time of the statements handling the data. In this study this tuning has been done intuitively rather than by experiment.

As a result however we can compute from the execution times of the different subsections of the programme that in those sections where execution time is not limited by the lineprinter speed the ratio between cpu time and drum transfer and access time lies between 5 and 10 indicating continuous work for the cpu.

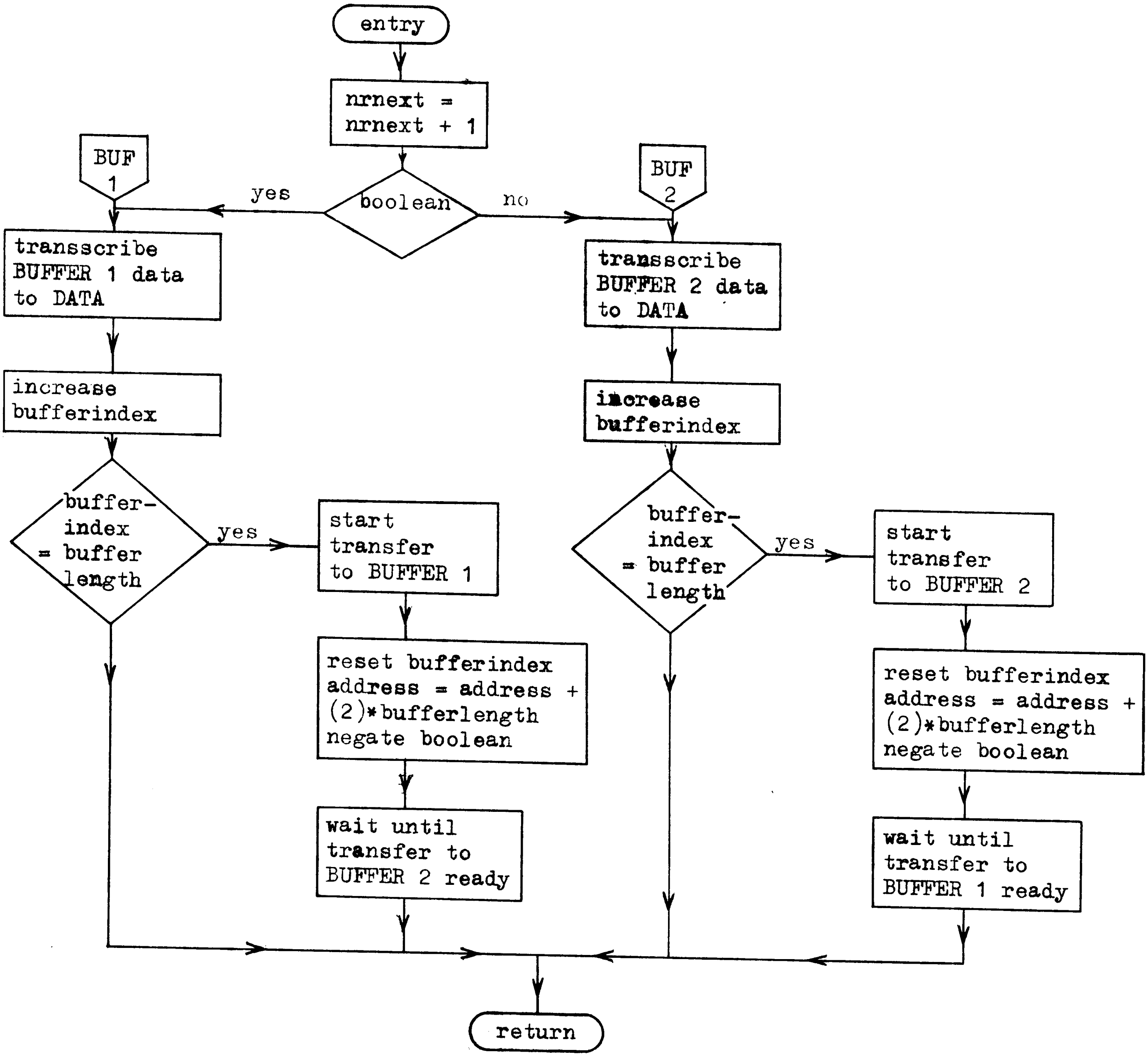

entry
nrnext = nrnext + 1
BUF 1
BUF 2
boolean
yes
no
transscribe BUFFER 1 data to DATA
transscribe BUFFER 2 data to DATA
increase bufferindex
increase bufferindex
buffer-index = buffer length
yes
start transfer to BUFFER 1
reset bufferindex address = address + (2)*bufferlength negate boolean
wait until transfer to BUFFER 2 ready
buffer-index = buffer length
yes
start transfer to BUFFER 2
reset bufferindex address = address + (2)*bufferlength negate boolean
wait until transfer to BUFFER 1 ready
return

Fig. 2-2 Outline of subroutine to read data from a file.

It will be clear at this point that output to a file is done very much along the same lines. The initialization is simpler and at the end of the file a transfer must be forced to empty the buffers. Details can be found in the appendices where the different buffer routines are discussed.

2.2. Layout of data on temporary (drum) storage during the DATA-DECODER and DATA-COMPUTATION programmes.

The general arrangement of the data on drum-storage during the execution of the DATA-DECODER and DATA-COMPUTATION programs is shown in Table 2-I. Especially in the sections which are used to hand over input information, calibration data and decoding results from the first to the second program, spare locations have been deliberately interspersed to accommodate future developments. A detailed description of the contents of this area is given in Appendix B.

The space for the current meter data per se has been calculated on the basis of sea endurance of our present current meters. At a rate of one registration every ten minutes the endurance is about 8 weeks, although we do have a registration of 11 weeks.

Space has been allocated for a maximum of nearly 15 weeks(the above registration rate gives 1008 samples per week so that a period of 14 weeks and 6 days can be dealt with).

The 8 track inputtape is read into the buffer area "INPUTTAPE" at the maximum speed of the paper tape reader. This is done to avoid errors due to slip when stopping the tape in the case that a more direct treatment of the data is used.

The results of the decoding is stored in the "DATA" area in the order: reference number, temperature, direction, velocity, errortag (see section 3). When a registration of an Aanderaa current meter is processed the additional salinity and depth data are stored in that order in the "SDDATA" area.

Further tests and computations are done on these files.

The results of the testing during the DATA-DECODER program are stored in the error tag word of the data-points under test. There is only one exception to this practice in that the end of whole registration and -possibly- detected clock stops are signalled by giving the reference number in the next data-points a value of 999999.

The program DATA-COMPUTATION has as a first task to build up the ERROR-TABLE (cf.Table 2-I) by scanning the error tags until this refno value 999999 is encountered. The ERROR-TABLE is then used in the subsequent treatment of the errors (see section 4.1).

The file HOURLY MEANS is build up during the calculation and printing of the individual data-points in DATA-COMPUTATION (cf. section 4.2 and Appendix E).

As can be taken from Table 2-I in the present set up a maximum of 300000 words of drum storage must be available for the program. Some -although rather crude- safeguards have already been built into the program to prevent the reading or writing in areas far outside the file-limits indicated in Table 2-I. As the use of some files is closely interconnected not all of the buffer routines need such a saveguard. As an example the files DATA, SDDATA and INPUTTAPE have a common saveguard in the buffer-routine for DATA (procedure DUMPMEET, cf.Appendix D).

2.3. Layout of data on temporary (drum) storage during the ACM program.

In this program two files are used: DATA and CORRECT, cf. Table 2-II. The program starts by buffering the paper inputtape (which is the output papertape of DATA-COMPUTATION) in the DATA area. (In fact all numbers are multiplied by 10 to save storage space). Then the corrections are inserted in the file in the proper place (see section 5.2).

Taking as the time step, the total elapsed time divided by the number of data cycles less one, which time step might be slightly different from 10 minutes, the time corresponding to the data points is computed, and the current components are corrected. These three data are then added to the CORRECT file in the order indicated in Table 2-II. Subsequent data products in the ACM program are derived from the file CORRECT.

TABLE 2-I.

Layout of data on temporary (drum) storage during the DATA-DECODER and DATA-COMPUTATION program.

used in		address	limits		array	array	
DECODER	COMPUTATION	lower	upper	filename	name(s)	type	contents/description
X	X	0	24	-	ADMIN	integer	initialization data, constants
X	X	25	39	-	CYCLESTORE	integer	results of decoding and testing concerning clock errors and length of data record
X	X	40	89	-	CALIB	real	calibration and initialization data
+	X	90	107	-	COMPASS	real	compass deviation data
X	-	200	max 75199	DATA	AT, BT, BUFONE, BUFTWO	integer	results of decoding and testing grouped per data points in the order: reference number, temperature, direction, speed, error tag
-	X	200	max 75199	DATA	DATA, WIG, WAG	integer	
X	X	80000	max 109999	SDDATA	ASD, BSD	integer	results of decoding for Aanderaa current meter per datapoints in the order: salinity, depth
X	-	120000	max 300000	INPUTTAPE	BUFONE, BUFTWO	integer	bufferstorage for 8-track input paper tape
-	X	120000	max 135000	ERRORTABLE	ERROR	integer	build up area for errortable for the interpolation of erroneous data
-	X	140000	max 175000	HOURLY MEANS	UURW	real	build up area for hourly means table during the computation and printing of the datapoints

- not used X used + filled only

TABLE 2-II.

Layout of data on temporary (drum) storage during the ACM program.

address	limits	filename	array	array	contents/description
lower	upper		name(s)	type	
0	max 89999	CORRECT	ABUF, BBUF	real	corrected time and current meter data in the order: time, north component (cm/sec), east component (cm/sec).
100000	max 129999	DATA	WIG, WAG	integer	buffer and correction area for the input paper tape with current meter data in the order: east component (mm/seo), north component (mm/sec).

3. Data-Decoder-Multipurpose.

In this chapter a description is given of the different techniques used in DATA-DECODER and of the meaning of the various tests. The full program listing is given in Appendix D.

This section ends with an explanation of a typical output and a discussion of results obtained sofar and information on run time on the EL-X8 installation of this institute.

3.1. Initialization, type-selection and function-selection.

The program expects two punched paper tapes: the first with information on identification of the station, type of current meter, calibration data of the different sensors and the function(s) to be performed on the data. This first paper tape is 7-track flexowriter code and is unformatted.

The second tape is an 8-track paper tape and is produced by translation of the magnetic tape from the current meter. The structure of the data contained in the 8-track tape will be discussed in detail in section 3.2.

Now let us return to the first tape. Since in this institute two types of current meter are in use, viz. Plessey MO21 and Aanderaa Model 4, which have a different number of sensors, the initial tape will be slightly different. As can be seen in figs. C-1 and C-2 of Appendix C in the Aanderaa case four more constants are used as calibration data of the extra salinity and depth (pressure) sensors.

Apart from these 4 extra constants the initial tapes are identical.

There are only two places in the initial tape where predetermined values can be expected and therefore are suitable for an automatic test on the correctness of the tape. These placed are the type of the current meter (place no. 7 in the tape) and the function(s) to be performed (the last number in the tape). If the tests fail, i.e. if the values do not correspond with what must be expected, the program will give a printer message and will terminate without any further action.

The first selection concerns the type of the current meter. This will either be Plessey or Aanderaa. A further complication arises from the fact that at a certain phase of the development of this program the translator was changed in such a way that the reference pulse (cf. section 3.2) is punched in track 8 and the parity bit in track 7 instead of the original

7 and 8 respectively. This has been done in order to facilitate visual inspection of the tape in case the program runs into trouble (cf. section 3.2).

To be able, however, to eventually reprocess the older tapes without retranslation, the program should provide for both kinds. This has been achieved by introducing extra information in the number representing the type. The value to be inserted in the initial tape is selected from Table 3-I.

The program DATA-DECODER has been devised to serve a number of purposes which have in common a step for decoding and format test of the 8-track tape. The subsequent function(s) to be performed will be selected by the value of the last number on the initial (7-track) input tape. Table 3-II shows the functions implemented in the programsas given in Appendices D and E.

Inspection of Table 3-II reveals that function number = 1 conforms to the standard processing, while function number = 0 will seldom be used.

Since some current meters have not been fitted with a temperature sensor function number = 2 is the standard processing in such a case.

Function number = 3 is meant to display the data from a laboratory test of an instrument for servicing purposes. It should be noted that function number = 3 is signalled to DATA-COMPUTATION as 0.

Function number = 9 is used to get the raw data printed for those registrations where automatic processing gives unsatisfactory results and visual inspection is needed to get a better interpretation. The raw salinity and depth data of an Aanderaa meter are not printed.

Since the printing can be done by a part of DATA-DECODER no subsequent program is needed for this printing.

The function numbers 4-8 have not yet been assigned specific functions and can therefore be used in future extensions. Just before termination the program DATA-DECODER puts a question for the successor program -if any- on the operator's command teleprinter (COTEL), who will take care to have that program in core.

In other installations a successive program might for instance be called from some library without the operator's intervention.

3.2. Decoding and format tests.

The Plessey current meter is fitted with 4 data channels. During a measuring cycle each channel is digitized to give a 10-bit word which is recorded on magnetic tape. At the end of the measuring cycle an extra

TABLE 3-I.

Selection of type-value		
	Plessey M021	Aanderaa Model 4
Reference in track 7 Parity bit in track 8	11	21
Reference in track 8 Parity bit in track 7	12	22

<u>TABLE 3-II.</u>

Functions implemented in the DATA-programs of Appendices D and E.

Function number	Test performed in DATA-DECODER			Other functions by DATA-DECODER	Subsequent program	DATA-COMPUTATION functions		
	temperature	speed	direction			interpolation	outputtape	
0	yes	yes	yes	-	DATA-COMPUTATION	all errors found	no	
1	yes	yes	yes	-	"	"	yes	
2	no	yes	yes	-	"	"	yes	
3	no	no	no	-	"	"gap readings" only	no	
4 5 6 7 8	TO BE DETERMINED		(SPARE)					
9	no	no	no	print of all data as after decoding but before computation, except salinity and depth data for Aanderaa	none	NOT APPLICABLE		

reference pulse is recorded as a delimiter.

These four channels are:

- a reference resistance, giving the identification number of the current meter
- temperature
- compass
- speed

The Aanderaa current meter is fitted with 6 data channels, comprising the same functions of the Plessey plus sensors for salinity and pressure (depth). The registration on the magnetic tape is completely the same: 6 10-bit words with a delimiting reference pulse.

The tape-translator reads these magnetic tapes and performs the following operations:

- a 10-bit word is divided in two groups of 5 bits which are stored as the bits 1-5 of two 8-bit characters.
- each group is inspected whether it are all zero's or not. If all zero's a sixth bit is set in that character otherwise the sixth bit is reset.
- if a reference pulse has been sensed immediately preceding the 10-bit word under consideration the eigth (reference) bit is set in the higher order character,otherwise reset. The reference bit in the lower order character is always reset.
- from each character thus obtained the parity is made even by setting or resetting the seventh (parity) bit.
- the two characters are fed to a paper tape punch, the higher order character first.

Fig. 3-1 shows part of the 8-track tape thus produced for a Plessey current meter. Except for the 4 extra characters (for the salinity and pressure channels) an Aanderaa tape is identical.
Fig. 3-1 clearly defines what structure should be expected in the character string constituting the registration.

A number of errors can however occur:

- malfunctioning of the registration part in the current meter may cause less than 10 bits in a word being written or words being "hidden" by other words written over it.
- improper adjustment of the translator can cause spurious reference bits being punched.
- logic errors in the translator or paper tape punch can give rise to illegal characters, i.e. true zero bits erroneously set or the parity bit having the wrong value.

The first two errors disrupt the proper structure of the character string, so that special actions are needed to get the most correct interpretation. The logic errors resulting in illegal characters are easier to cope with in that e.g. a simple interpolation in the pertaining measuring channel will give a most apparent value.

The structure of the character string can be formally defined as shown in Table 3-III where the well-known Backus-Naur form (BNF) has been used.*)

From Table 3-III it can be understood that the definitions of the Plessey and Aanderaa registrations are given with the decoding process in mind rather than the actual registration process, where new data cycles are added at the end of the already existing registration. In general such so-called right-recursive definitions as used here are considered to have certain advantages relative to left-recursive constructions when the structure analysis is considered as a syntax analysis (see e.g. Foster, 1970).

It must be emphasized however that the definitions in Table 3-III are not fully context-free in the sense of the theory of formal languages. Clearly the translate error definitions could have been given as

<translate error> :: =

<registration error > <registration error>

with the additional constraint that a translate error must consist of the same number of characters as a data cycle.

With the definitions of Table 3-III at hand the decoding and structure analysis process is rather straight forward (see fig. 3-2).

To start the process the input character string is scanned for the first reference character, i.e. character where the reference bit is set.

*)

For the reader not familiar with the BNF notation:
the first two lines of Table 3-III should be read as:
- an <input string> is defined as a <registration> followed by a <terminal character>, and
- a <registration> is defined to be a <Plessey registration> or an <Aanderaa registration> .

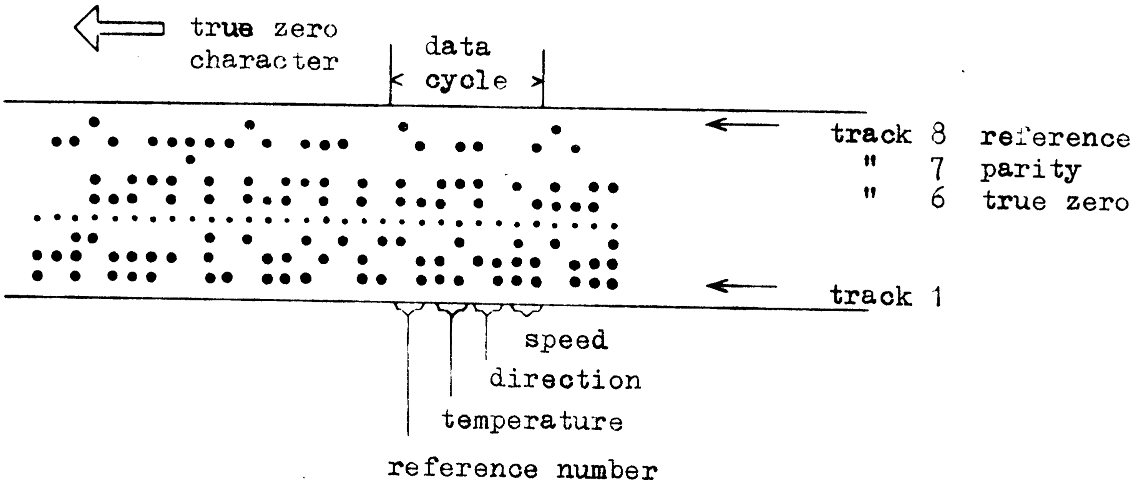


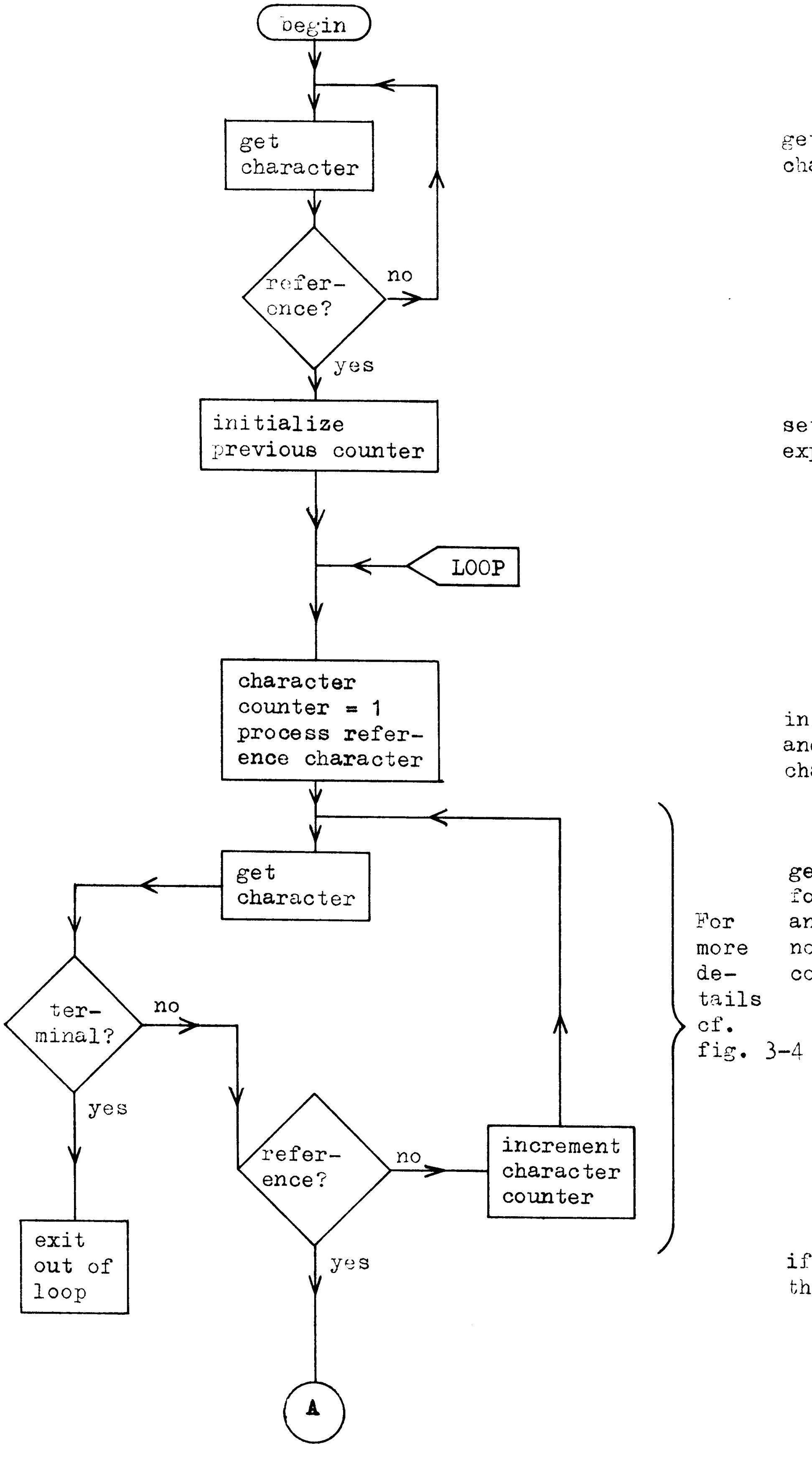
Fig. 3-1 Part of a 8-track papertape for a Plessey current meter.

TABLE 3-III.

Formal definition of the structure of the input character string

```
<input string> ::= <registration>  <terminal character>
<registration> ::= <Plessey registration> | <Aanderaa registration>
<Plessey registration> ::= <Plessey part>  <Plessey registration>
                           <Plessey part>
<Plessey part> ::= <Plessey cycle> | <Plessey translate error> |
                   <registration error>
<Aanderaa registration> ::= <Aanderaa part>  <Aanderaa registration>
                            <Aanderaa part>
<Aanderaa part> ::= <Aanderaa cycle> | <Aanderaa translate error> |
                    <registration error>
<Plessey cycle> ::= <reference>  <temperature>  <direction>  <velocity>
<Plessey translate error> ::= <reference>  <Plessey ref tail>
<Plessey ref tail> ::= <temperature>  <temp tail> |
                       <reference>  <direction>  <velocity>
<Aanderaa cycle> ::= <reference>  <temperature>  <salinity>  <pressure>
                     <direction>  <velocity>
<Aanderaa translate error> ::= <reference>  <Aanderaa ref tail>
<Aanderaa ref tail> ::= <temperature>  <Aanderaa temp tail> |
                        <reference>  <salinity>  <pressure>  <direction>
                        <velocity>
<Aanderaa temp tail> ::= <salinity>  <Aanderaa sal tail> |
                         <reference>  <pressure>  <direction>  <velocity>
<Aanderaa sal tail> ::= <pressure>  <temp tail> |
                        <reference>  <direction>  <velocity>
<temp tail> ::= <direction>  <reference> | <reference>  <velocity>
<temperature> ::= <channel>
<salinity> ::= <channel>
<pressure> ::= <channel>
<direction> ::= <channel>
<velocity> ::= <channel>
<reference> ::= <reference character>  <character>
<channel> ::= <character>  <character>
<registration error> ::= <reference character>  <error tail>
<error tail> ::= <character>  <error tail> | <empty>
```

Fig. 3-2


begin
get character
refer- ence?
no
yes
initialize previous counter
LOOP
character counter = 1 process refer- ence character
get character
ter- minal?
no
yes
exit out of loop
refer- ence?
no
yes
increment character counter
A
For more de- tails cf. fig. 3-4

get first reference character

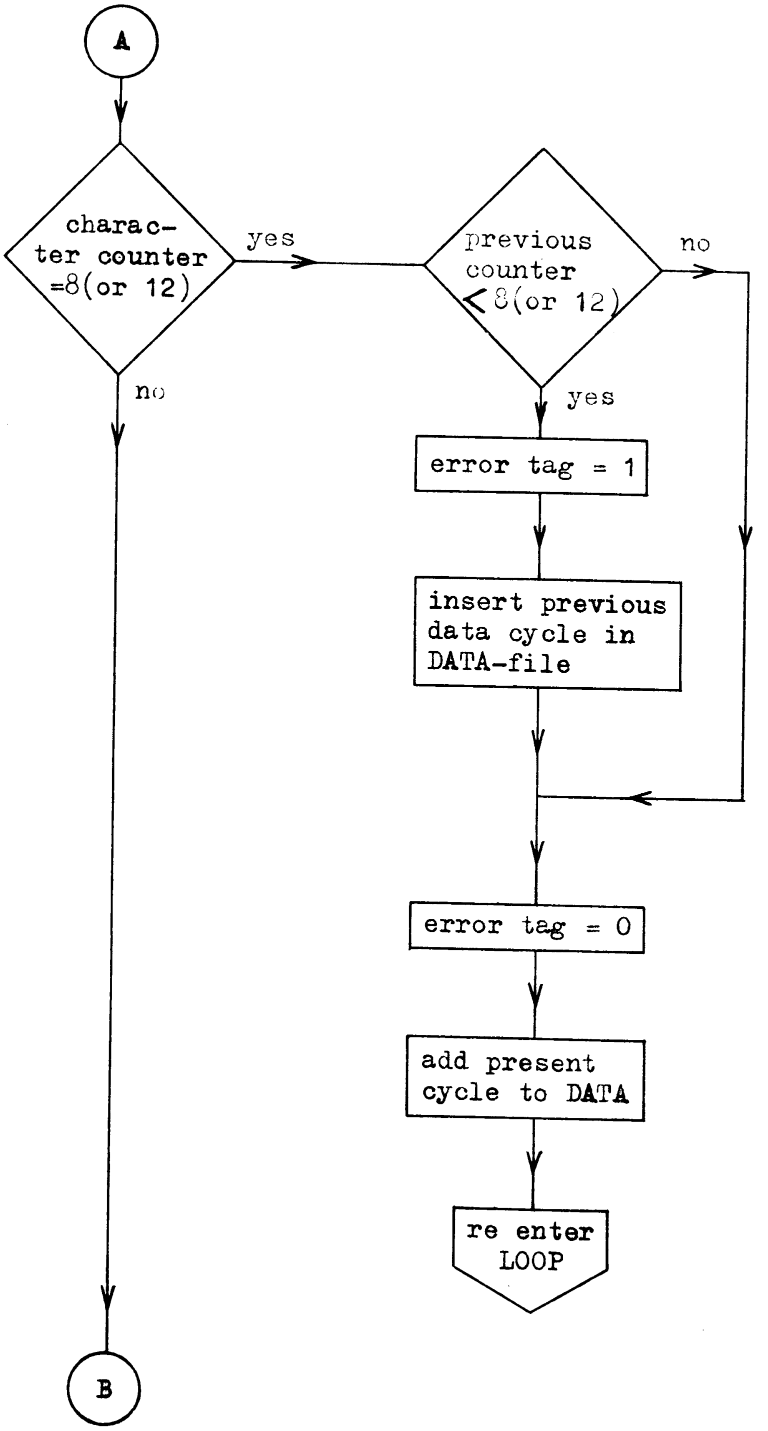
set "previous counter" to expected value (8 or 12)

initialize character counter and process reference character (parity check etc.)

get next character and test for terminal character and test for reference if not increment counter and continue the loop

if reference exit out of the loop

Fig. 3-2 continued 1.



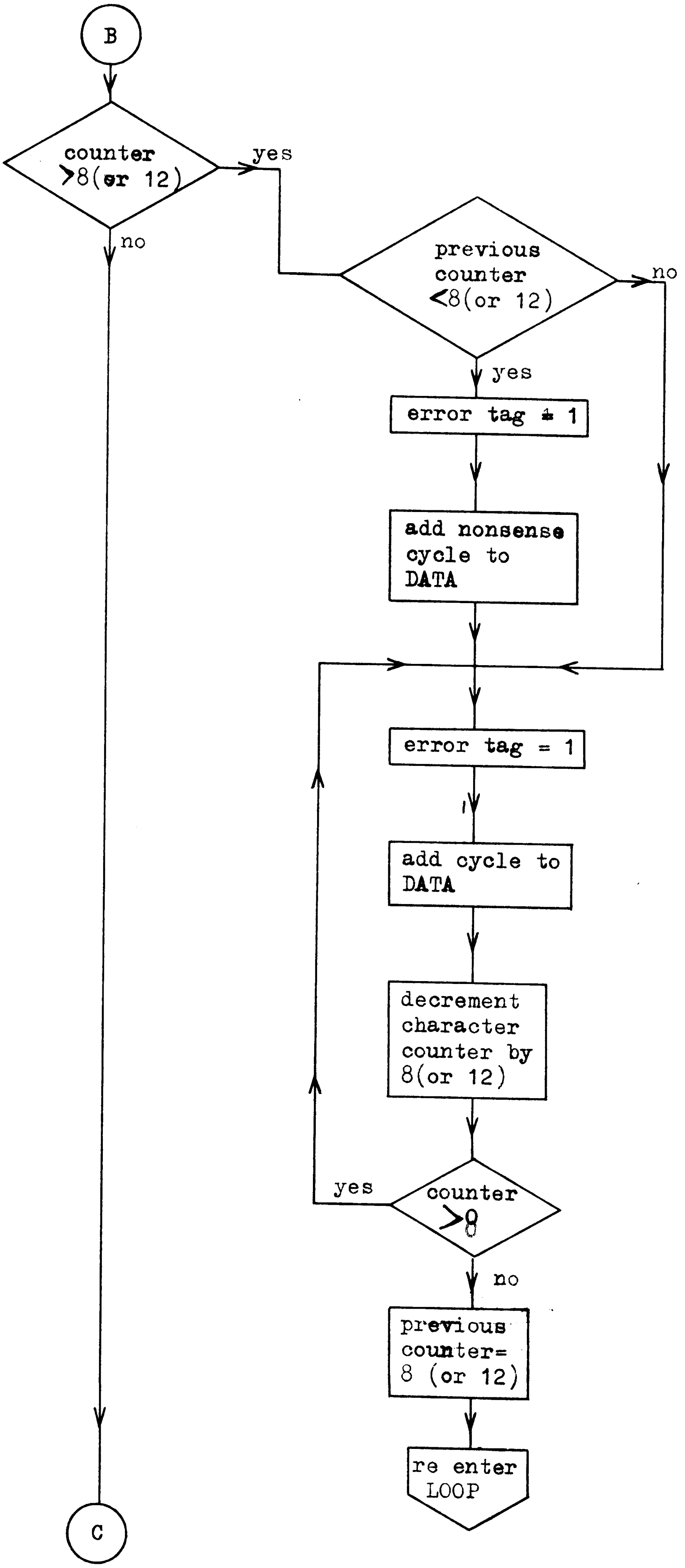
test character counter and
if correct check for previous
data-cycle complete

if previous not complete add
extra data-cycle with
errortag = 1

add present data-cycle to
DATA-file

and start loop for next
data-cycle

Fig. 3-2 continued 2.


B
counter >8(or 12)
yes
no
previous counter <8(or 12)
no
yes
error tag = 1
add nonsense cycle to DATA
error tag = 1
add cycle to DATA
decrement character counter by 8(or 12)
yes
counter >8
no
previous counter= 8 (or 12)
re enter LOOP
C

Note: counter 8 (or 12)

has present cycle more than 8 (or 12) characters? if so, was previous cycle incomplete? if yes,add it to DATA with errortag = 1

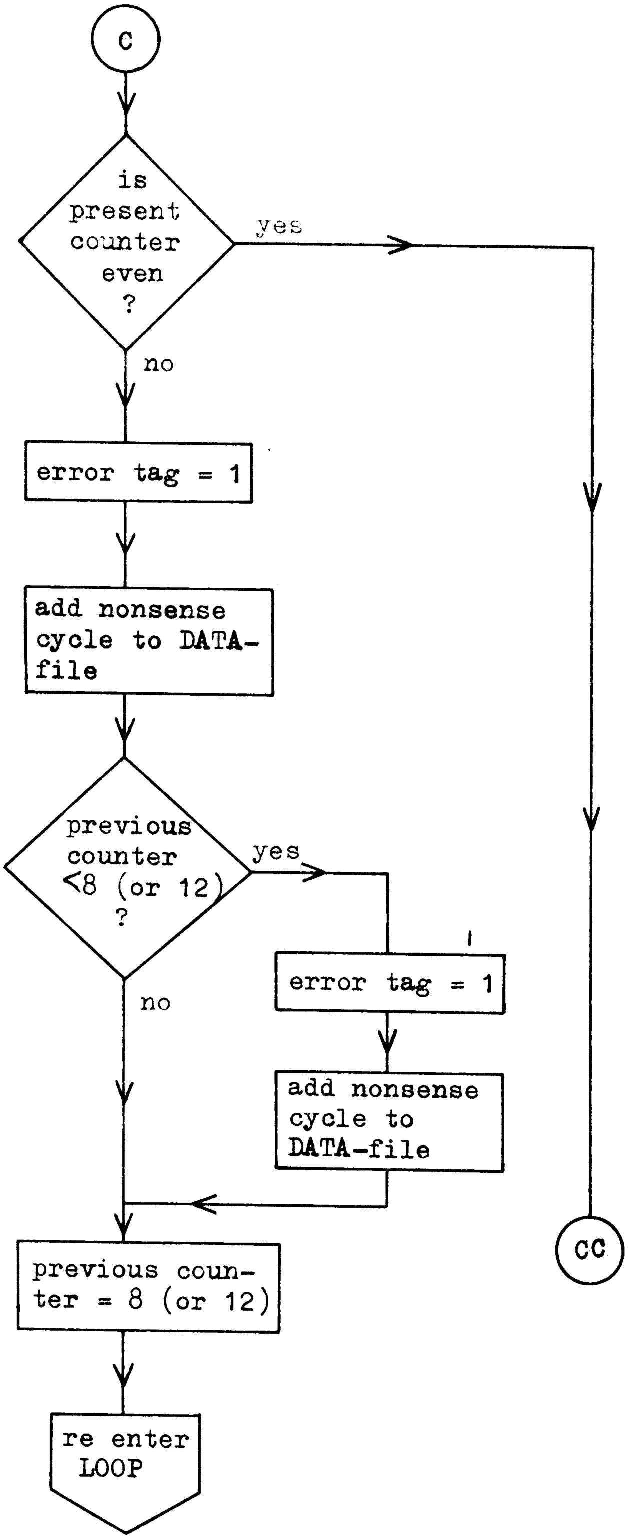
add a nonsense cycle to DATA with errortag = 1

subtract the correct number from character counter and test sign of remainder

if positive add another nonsense cycle to DATA

if negative start a new decoder cycle after resetting previous counter to expected value.

Fig. 3-2 continued 3.

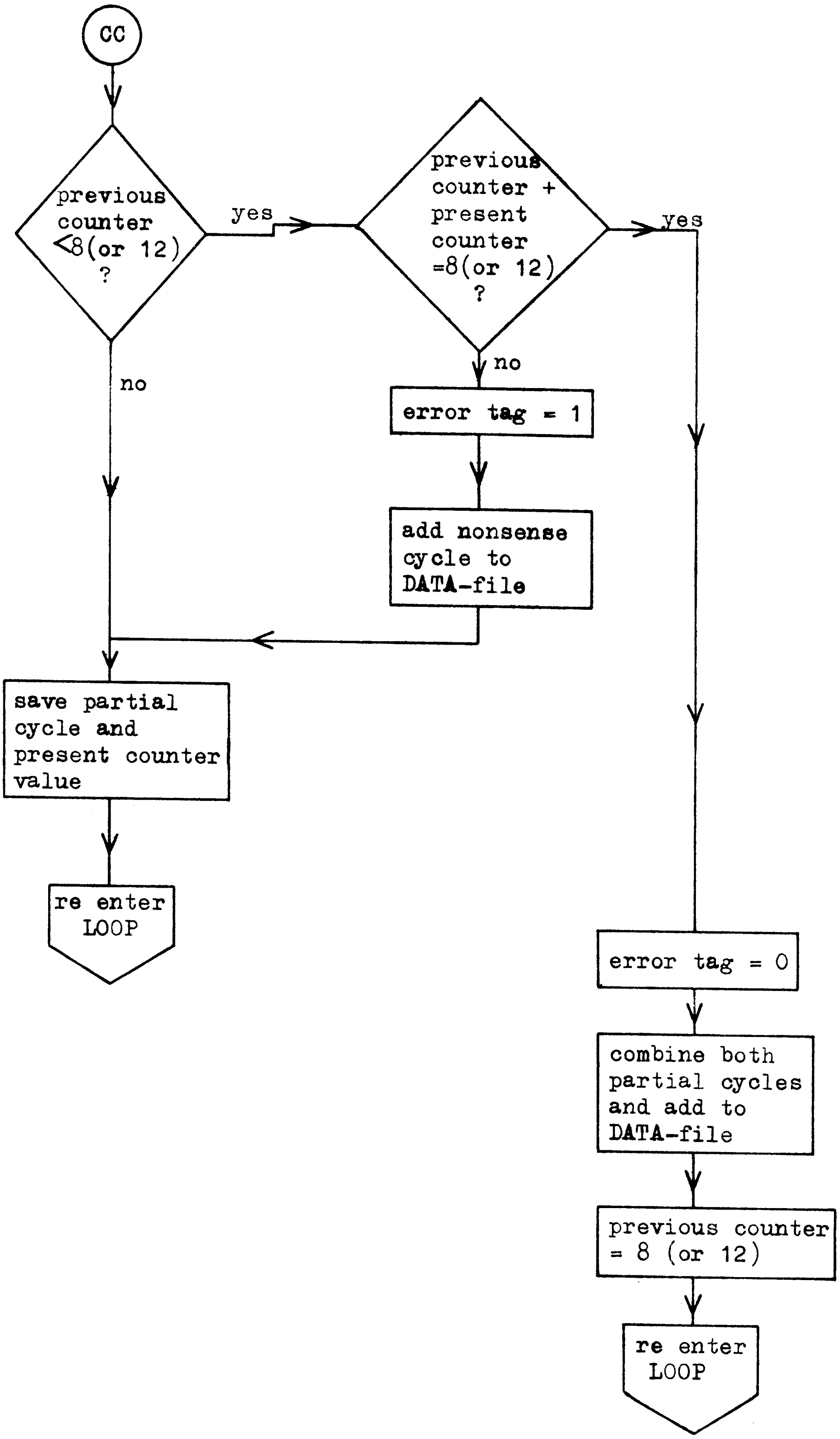

C
is present counter even ?
yes
no
error tag = 1
add nonsense cycle to DATA-file
previous counter <8 (or 12) ?
yes
no
error tag = 1
add nonsense cycle to DATA-file
previous counter = 8 (or 12)
re enter LOOP
CC

if the present counter value is odd no < translate error > can be involved so a < registration error > is the proper counclusion

if moreover the previous counter was not 8 (or 12) one more nonsense cycle must be filed

and reset previous counter to expected value

Fig. 3-2 continued 4.



CC
previous counter <8(or 12) ?
yes
no
previous counter + present counter =8(or 12) ?
yes
no
error tag = 1
add nonsense cycle to DATA-file
save partial cycle and present counter value
re enter LOOP
error tag = 0
combine both partial cycles and add to DATA-file
previous counter = 8 (or 12)
re enter LOOP

check previous counter

try a <translate error>
if test fails add nonsense cycle to DATA-file

save present partial cycle and character counter

and start next decoder cycle

if translate error recombine to proper cycle and add to DATA-file

reset previous counter to expected value
and
start next decoder cycle

Fig. 3-2 continued 5.

Fig. 3-2 Principles of flowchart for decoding and structure test of input character string.
(Note: in this flowchart some resetting of variables when re-entering LOOP is understood implicitly.)

Then the subsequent characters are converted to form the various data channels incrementing a counter after each character. As soon as the next reference character is sensed, the counter is tested for the corresponding value: 8 or 12 characters for Plessey and Aanderaa current meters respectively. If the test succeeds the previous counter value is tested. If that cycle was not complete a nonsense cycle with errortag = 1 is added to the DATA-file. Then the converted data of the present cycle are stored in the DATA file after adding a tag (cf. section 3.4 and 3.5). In the case of an Aanderaa the salinity and pressure data are saved in a separate file, SDDATA.

If however the test fails two different possibilities must be distinguished:

- the counter indicates more than 8 or 12 characters
- the counter indicates less than the expected number.

In the case of more than 8 (or 12) characters the previous counter value is tested for completeness of the previous cycle. If this value was less than 8 (or 12) a nonsence cycle is added to the DATA-file with the errortag = 1. Then subsequent nonsense cycles are added again with errortag = 1, decrementing the present counter by the appropriate value for each cycle as long as the counter value is positive.

By this procedure it is achieved that data cycles registered one over another on the magnetic tape will be interpreted as an appropriate number of cycles to (hopefully) maintain the correct timing of our data. Clearly the automatic fashion of decoding employed cannot cope with insufficient technical servicing of the current meters resulting in e.g. a "registration disaster".

When the counter value is found to be less than expected, a choice must be made between a <registration error> and a possible < translate error > . From the structure in Table 3-III it is clear that a < translate error > is possible only if the counter value is even, which possibility requires further inspection before being accepted. Otherwise a <registration error> is detected and properly taken care of. In that case possibly a previous incomplete cycle though with an even counter value is considered to be a < registration error > too.

When the possibility of a <translate error> must be considered the action depends on the value of the previous counter:

1. if this value is 8 (or 12) the present partial cycle is saved together with the present character counter for use in the next decoder cycle.

2. if the previous counter is less than 8 (or 12) and the sum of the previous and the present counters does not equal 8 (or 12) a nonsense cycle with errortag = 1 is added to the DATA file and the present partial cycle and the present character counter are saved for subsequent use.
3. if the sum of previous and present counters equals the expected value both partial cycles are combined to give a complete data cycle which is then added to the DATA file.

The previous counter value is reset to 8 (or 12) for use in future tests.

Again it must be emphasized that an improper adjustment of the translate machine, resulting in lots of illegal reference bits cannot be corrected by an automatic data analysis into a state of complete integrity of the data produced!

Finally note that, whatever functions were performed under control of the values of the present and previous character counters, the processing of the reference character, the occurrence of which caused these actions, is essentially part of the next decoder cycle.

The tests decribed sofar are concerned with the structure of the input string only. We will now turn our attention to the recognition of the different characters and the parity and true zero tests.

As mentioned before the input string consists of 8 bit characters which thus can have values from 0 through 255.

Two special characters can be distinguished beforehand: the "blank" with value 0 which has no significance and therefore can be ignored and the "terminal character" with the value 255, marking the end of the input string.

The remaining 254 characters have a meaning which depends on the place of the reference and parity bits in the character (cf. Table 3-I). In fig. 3-3^{a,b} the significance of the different bits in both cases are indicated.

According to the type of the current meter, i.e. either 11/21 or 12/22, a 256 element TRANSLATE-table is built up as follows:

- The n-th element is assigned the value n modulo 32.
- Then elements 33 through 63, 97 through 127, 161 through 191 and 225 through 255 are reassigned to have the value 50. These elements are selected to correspond to characters where the "true zero bit" is set although the value part of the character is nonzero.
- As a third step the elements corresponding to "reference characters", i.e. characters with the reference bit set are negated. This applies to the elements 128 through 255 for the reference bit in track 8, whereas the elements 64 through 127 and 162 through 255 are involved in the old tapes with track 7 for the reference bit.

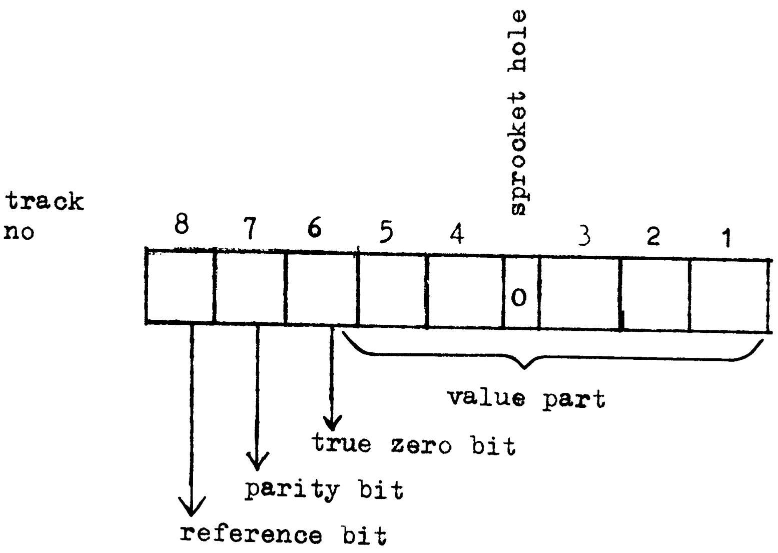


Fig. 3-3^a Significance of character bits for type = 12 and type = 22 current meter tapes; present situation.

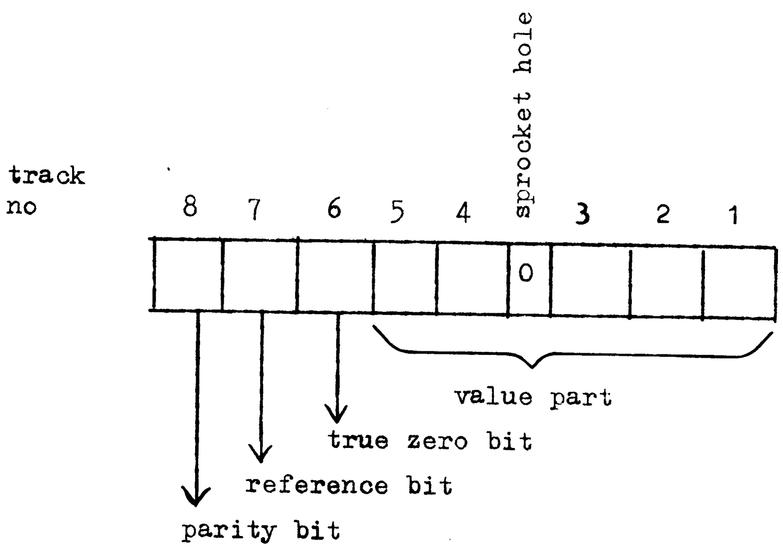


Fig. 3-3^b Significance of character bits for type = 11
and type = 21 current meter tapes; old situation.

- As a last step the elements 0 and 255 are reassigned the values -255 (blank) and 255 (terminal character) respectively. Note that the element 0 (blank) must be distinguishable from an element with value part = 0 and the true zero bit set.

A second table of 256 elements, the PARITY table, is filled with the boolean values "true" and "false" in such a way that the n-th element is "true" if the parity of the character corresponding to that element is odd, and "false" if the parity is even.

With these two tables at hand the character tests are performed as follows, cf. fig. 3-4.

The appropriate value for the next character is obtained by table look up in the TRANSLATE table at the element given by the value of a function "list". This function gives as its output the value of the next 8 bit character in the input string. As a side-effect the corresponding value of the PARITY-table element is assigned to the boolean "wrongparity".

As has been mentioned earlier, "blanks"
- indicated by character = -255- are ignored.
Then the character is tested for the terminal value 255. If this test succeeds the end of the input string is reached and execution will exit from the decoder loop (cf. fig. 3-2).

Otherwise the sign of "character" is sensed for the reference indication. If it is a reference character the checking of the present value of the character counter and eventually the value in the previous cycle starts, see fig. 3-2 and its explanation. The processing of the reference character itself is done at the beginning of the next decoder cycle. This processing involves:

- dropping the minus sign
- if wrongparity = true, i.e. the character has an odd parity a boolean "skip" is set "true" otherwise "false". This boolean governs the interpretation of the characters before data are added to the DATA-file.
- if character = 50 indicating an illegal true zero bit "skip" is set "true"
- the value of "character" is saved in "number" for subsequent use
- the character counter is reset to 1.

Then execution will continue at GETCHAR.
If no special character (blank, terminal or reference) is found the character is treated in the standard way:

- the character counter is incremented by 1
- if wrongparity = true "skip" is set "true"

- if an illegal true zero bit is detected "skip" is set "true" also
- if the character counter is even and skip = false the value of "number" and "character" are combined and stored in the proper place of the data cycle. If, however, skip = true the action depends on the channel involved:
  a. for reference, temperature, salinity and pressure the previous value is inserted as the present one
  b. for the direction and velocity channels a 0 is inserted, which at a later stage will enforce an interpolation of the value from the surrounding data (cf. section 4.1).
  Finally "skip" will be set "false"
  c. if the character counter is odd the value of "character" will be saved in "number" for later use.
  In either case execution continues at GETCHAR.

Apart from the structure and character tests same more checks are part of the decoder cycle.

Two of these checks are concerned with the direction and velocity channels and are performed by the routine which adds a data cycle to the DATA file. A detailed discussion of these checks will be given in the sections 3.4 and 3.5 describing the tests on the velocity and direction channels respectively.*)

3.3. Test for temperature channel.

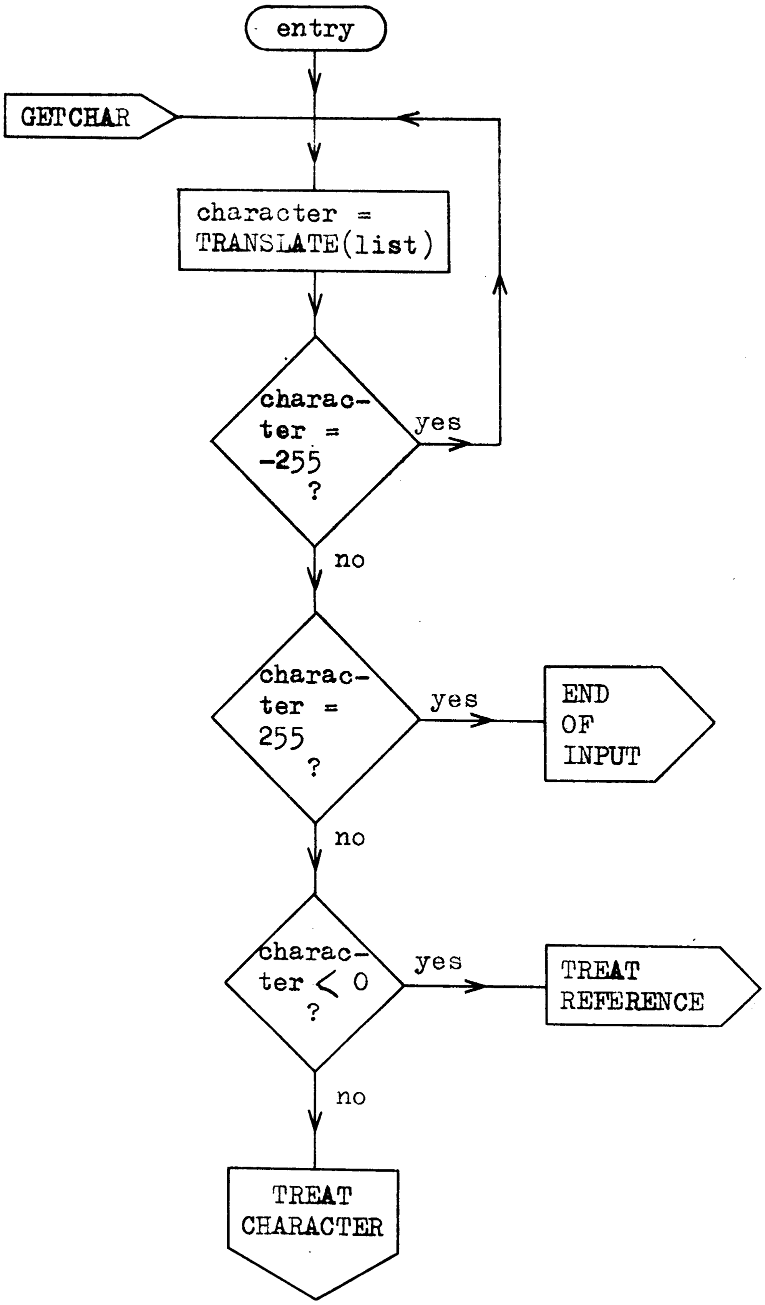
The temperature channel, if a sensor was installed on the pertaining current meter, is in our setup used for two purposes:
- as a mark of the begin- and end times of the periode that the instrument was moored
- as a temperature measurement proper.

The test described in this section is to serve the first purpose only. A test on the temperature data as such is not performed. As a result of these measurements only hourly means are computed neglecting values that are obviously "too far off" of the previous mean value.

The search for the first and the last measurement with the mooring set is based on the fact that temperature fluctuations in the Southern Bight of the North Sea are virtually nonexistent. A weak tidal signal with amplitudes of the order 0.1 - 0.3 degrees only scarcely occurs on top of a long term gradient which in spring and fall can reach peak values of upto 2°C/week but is much less during the rest of the year.

*)One more test, involving the temperature channel, is part of the decoder cycle. A description is given in section 3.3.

Fig. 3-4



fetch the next character

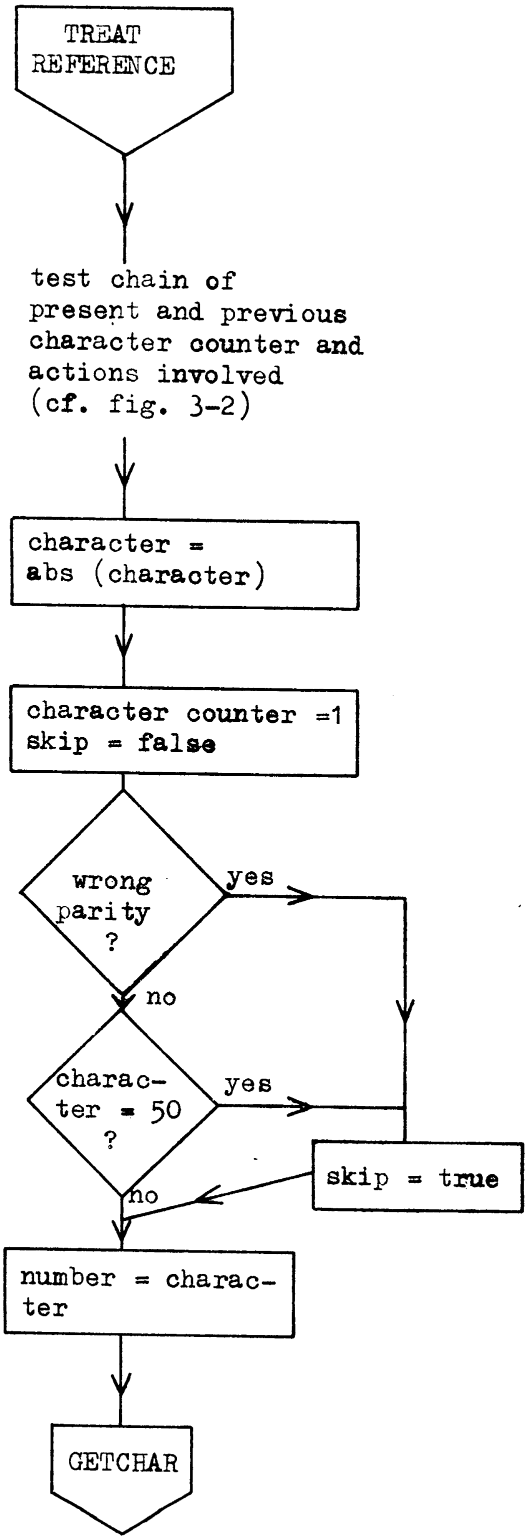
if it is a blank then ignore

if it is a terminal character exit to a proper continuation label

if it is a reference character perform the required actions

else treat the character

Fig. 3-4 continued 1.



perform the tests and actions required as shown in fig. 3-2

drop minus sign

reset character counter to 1 and (re)set skip to false

if wrong parity

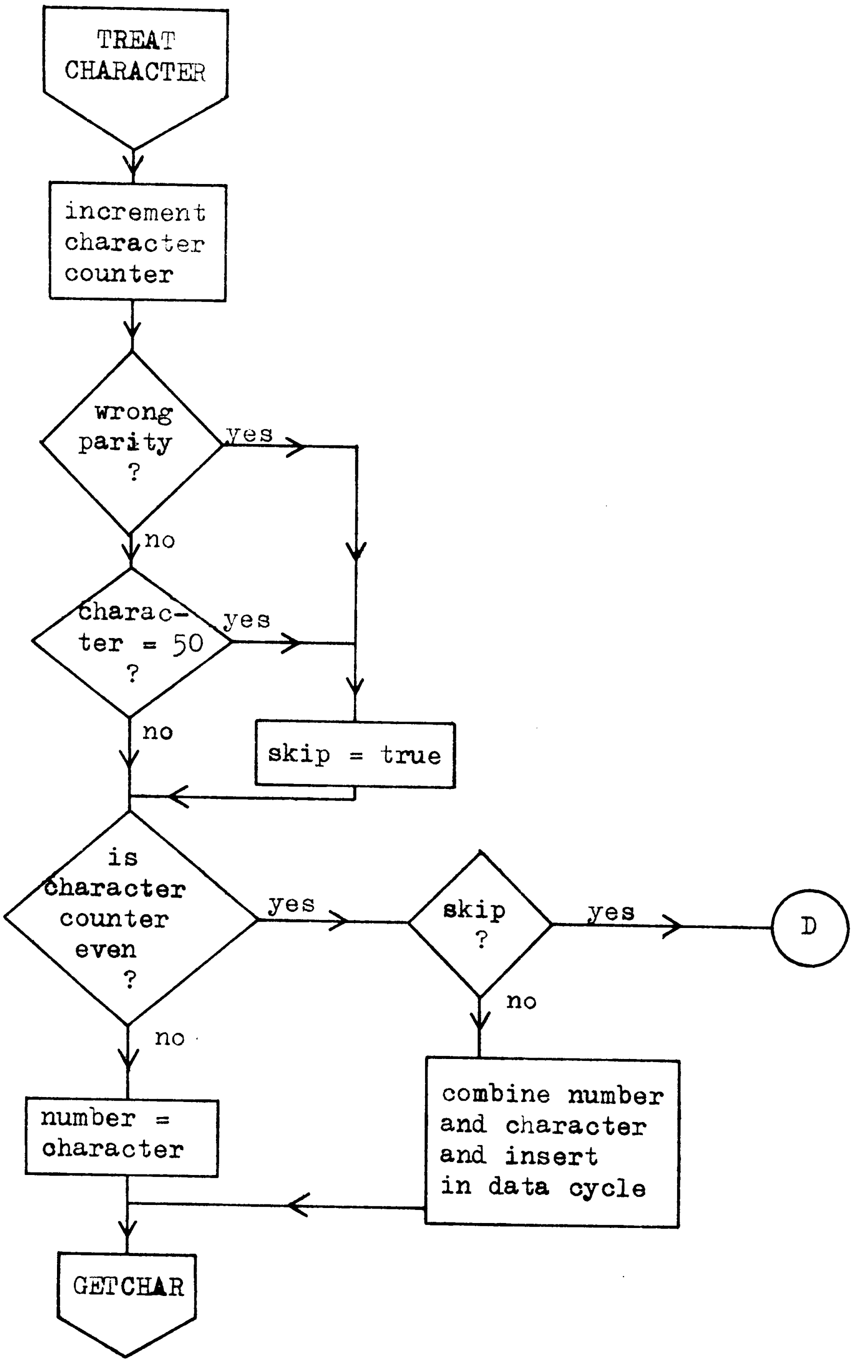
and/or illegal true zero bit

set skip = true

save character

and fetch next character

Fig. 3-4 continued 2.



increment character counter

if wrong parity

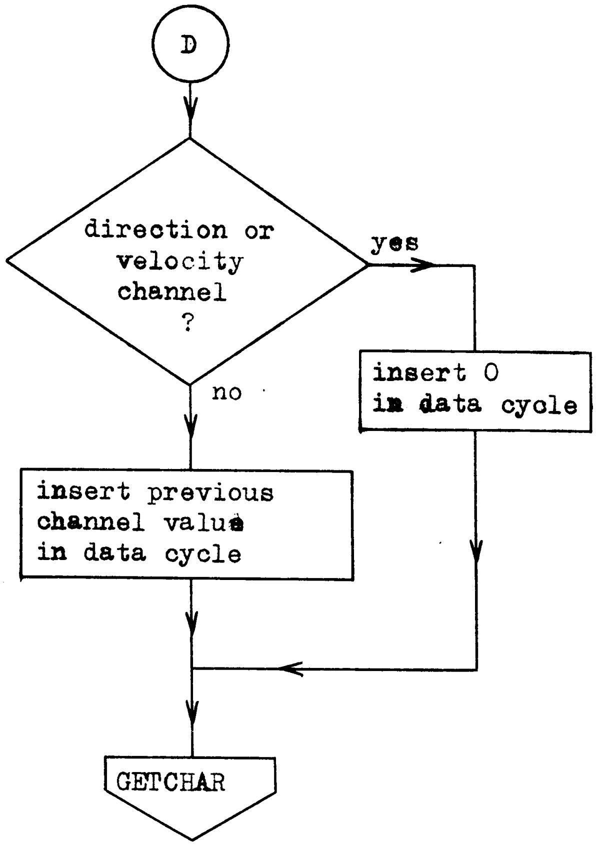
and/or illegal true zero bit

set skip = true

if character counter is odd save character and fetch next chracter

if character counter is even test skip if false combine number and character and insert in data cycle

Fig. 3-4 continued 3


D
direction or velocity channel ?
yes
no
insert 0 in data cycle
insert previous channel value in data cycle
GETCHAR

if the channel to be skipped is direction or velocity then insert 0 in data cycle otherwise previous value

Fig. 3-4 Character recognition and processing flowchart. Detail of fig. 3-2.

Therefore the start of the mooring period is assumed as soon as the differences between three successive hourly mean values of the temperature readings are less than 3, which depending on the calibration of the thermistor amounts to 0.1 - 0.2 degrees.*)

The average value thus obtained is used to scan the individual measurements until the first measurement that fits the above criterion of less than 3 units difference. This measurement is subsequently considered to be the first measurement cycle with the instrument moored at its target depth.

The search for the last measurement evolves along much the same lines.

The temperature channel of the whole registration is scanned, comparing each individual value with an hourly mean value that is updated during te scan. As soon as three successive datacycles show a difference of more than 10 units (i.e. 0.3 - 0.7 degrees) the end of the useful registration periode is assumed and indicated as such in the DATA file.

Clearly improper functioning of the digitizer in the current meter can yield an erroneous result in this test, but the registration would then be of limited value anyway!

In our experience the criteria described above give results which coincide with results from visual inspection combined with data from the mooring information sheet.

Clearly other sea areas might enforce different values in the criteria.

No correction of erroneous temperature data has been envisaged, but an implementation of such a feature is possible in a later stage.

3.4. Test for the speed channel.

As an introduction of this section we will briefly discuss the speed measurement.

The movement of the water relative to the current meter is transformed in a rotation of the propeller (Plessey) or the rotor (Aanderaa). By a magnetic coupling this rotation is fed to a reducing gearbox which in turn advances the slide contact of a potentiometer. During a registration cycle a digital resistance value equivalent to the position of the slider is registered on the magnetic tape.

*) The computation and tests of these hourly means are performed during decoder part of the program.

A current speed is thus translated in a difference of the resistance values, occurring in the registration.

Both current meter types have the problem of a "gap" in the potentiometer and the possibility of erroneous digitizing in common. The Plessey meter has one more check point in that damage or removal of the vane will couse the propellor to be no longer at the front end but at the rear end of the instrument. The propeller will then rotate in the wrong direction which is detectable in the sign of the changes of the subsequently registered resistance values("a negative speed" results).

As already indicated, the "gap" in the potentiometer calls for special attention. The "high" end and the "low" end connections of the potentiometer have a certain dimension as has the isolation between them. The "gap", spanning the three areas implicated above, will show up in the registration as a resistance value of 0, 1, 1023 (all ten bits set) or some fancy number in between, depending on the particular current meter. As the numbers 0, 1 and 1023 cannot occur outside the "gap" they can be recognized as such and indicated in the DATA file. This signalling is done during decoding the input-string by the routine which adds a data cycle to the DATA file. A speed gap, which is considered a special kind of speed error, is indicated by giving the errortag (cf. section 3.2) the value 100 (see also Table 3-IV).

The nonsense numbers generated by some current meters in the "gap" can in general only be detected as erroneous digitizing or "suspect data" in the rest of the speed tests.

As will be clear from the above these tests are:

- detection of "suspect data"
- "negative speed" detection in the Plessey case.

Detection of "suspect data". The criterion for this test must be able to deal with fully different current situations.

In the Southern Bight the current is mainly of tidal origin (amplitudes off the Dutch coast are up to about 1 m/sec) combined with a small residual current (typical order of 5 cm/sec).

Moreover the tidal current can vary from a purely alternating current to a nearly constant but rotating current characteristic for an amphidromic point. After some experimenting we reached at a fully empirical criterion which gives the same results as a most critical visual inspection could give us. (It should be noted already here that results mentioned incorporate the interpolation procedure described in section 4.1. This means that possibly shortcomings of the speed criterion though rarely occurring, can be corrected with a good deal of success by the interpolation procedure used.)

TABLE 3-IV.

The values of the errortag.

Value	Meaning
0	complete data cycle, needing no special treatment of the direction and speed channels.
1	incomplete data cycle, to be recovered by interpolation of all channels in the subsequent analysis (cf. section 3.2).
10	complete data cycle; direction channel needs special treatment due to either gap or suspect value (cf. section 3-5).
100	complete data cycle; speed channel needs special attention due to either gap or suspect value (cf. section 3-4).
110	complete data cycle; both direction and speed channels need special processing due to gap and/or suspect values.

The criterion consists of two parts and can be described as: "a speed reading is considered "suspect" when it differs from the preceding value by more than 10 cm/sec and by more than 20% of that value".

The criterion is graphically displayed in fig. 3-5.
This criterion -implemented as a boolean function- is now used as the central building-stone of the test of the speed channel (see fig. 3-7).

This test comprises essentially five levels when the initializing has been done:

1. test the channel reading as compared with its immediate non-gap, non-suspect predecessor called "previous value".
2. if this test fails try the next non-gap reading as compared to "previous value".
3. if this test fails too, try the predecessor of "previous value". If this test is succesful "previous value" is considered "suspect" as yet and is tagged as such.
4. if this test fails too a third reading is taken into consideration. If these three values "fit to each other" there remain two possibilities:
   a. it is a rapid change of speed before/after the turn of the tide
   b. the clock of the current meter stopped, to restart again at a later time.

   The first possibility is readily detectable by looking at the time dependence of the five speed channel readings involved sofar: "previous value", its predecessor and the three subsequent readings. If the conclusion cannot be alternative a. the second alternative called "clock-failure" is chosen.
   This is indicated in the DATA file as an end of file mark and by introducing the number of the data cycle in a separate table.
   The initializing for the test of the rest of the record has essentially been done in that the three first readings fit each other.
5. If also the test at level 4 failed a maximum of two more channel reading is tried to fit "previous value". If again this test fails a "clock-failure" is assumed and subsequently marked as above.
   In this case however re-initialization of the test sequence is needed as the first three readings do not fit among each other.

These five levels of testing yield the schematic flowchart given in fig. 3-7.

The boolean function, which constitutes the test criterion, also tests for the proper sequence of the speed channel readings in the Plessey case. As soon as the sign of the changes is stably negative (subsequent resistance values are increasing numbers except when the gap is in between) the tail fin is assumed to be lost and further testing is stopped. In the DATA-file an end-of-file mark is inserted.

From the above explanation it can be understood that a clock failure in principle can occur an apriori unknown number of times. In the program this number originally was arbitrarily limited to ten, although such a high number can already be considered as making the registration worthless for further treatment.

In practice however we found that under extreme condition (i.e. for the Southern Bight extreme) the criterion sometimes fails and indicates a clock failure where according to visual inspection there is none. Therefore we left the original limit of 10 unchanged as an extra safety margin.

At only one occasion the computer beyond a certain point of the program produced complete nonsense until the operator stopped the machine. A distinct reason could not be identified. The current meter concerned had been moored on one of the banks inside the Frisian isles and fell dry part of the tidal period. Moreover the propellor was stuck a couple of times by sea-weed!

The around 90 remaining registrations processed so far indicate that the criterion, although entirely empirical, works to any manually feasible level of accuracy in the data produced.

3.5. Test for the direction channel.

Quite analogous to the speed measurement, the measurement of the current direction is implemented as a resistance measurement. A contact wire, which moves with the compass needle, is drawn against a potentiometer as the "sliding contact" at measuring time.

Thus we have also the "gap-problem" in the direction channel. This again gives rise to the gap-readings 0,1 or 1023 (all ten bits set) or (depending on the current meter) a nonsense number.

The gap-readings are marked by the routine which adds the data cycle to the DATA-file. In accordance with Table 3-IV direction-errors whether "gap" or "suspect" are signalled in the appropriate errortag by incrementing this tag by 10.

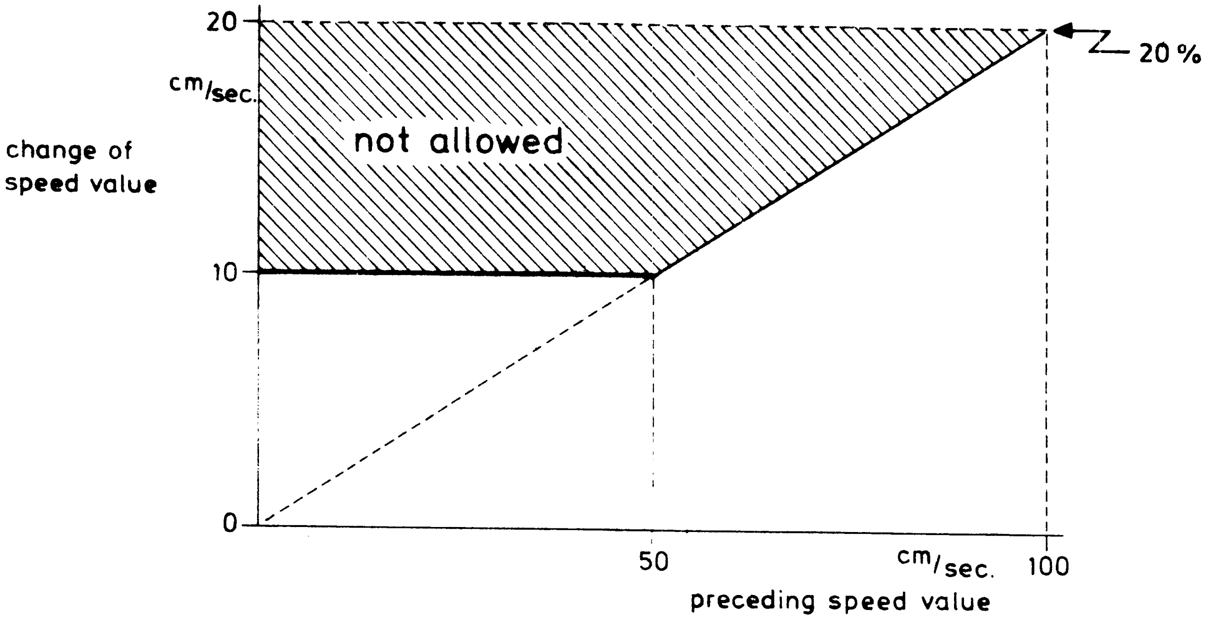


Fig. 3-5: Graphical representation of the speed criterion

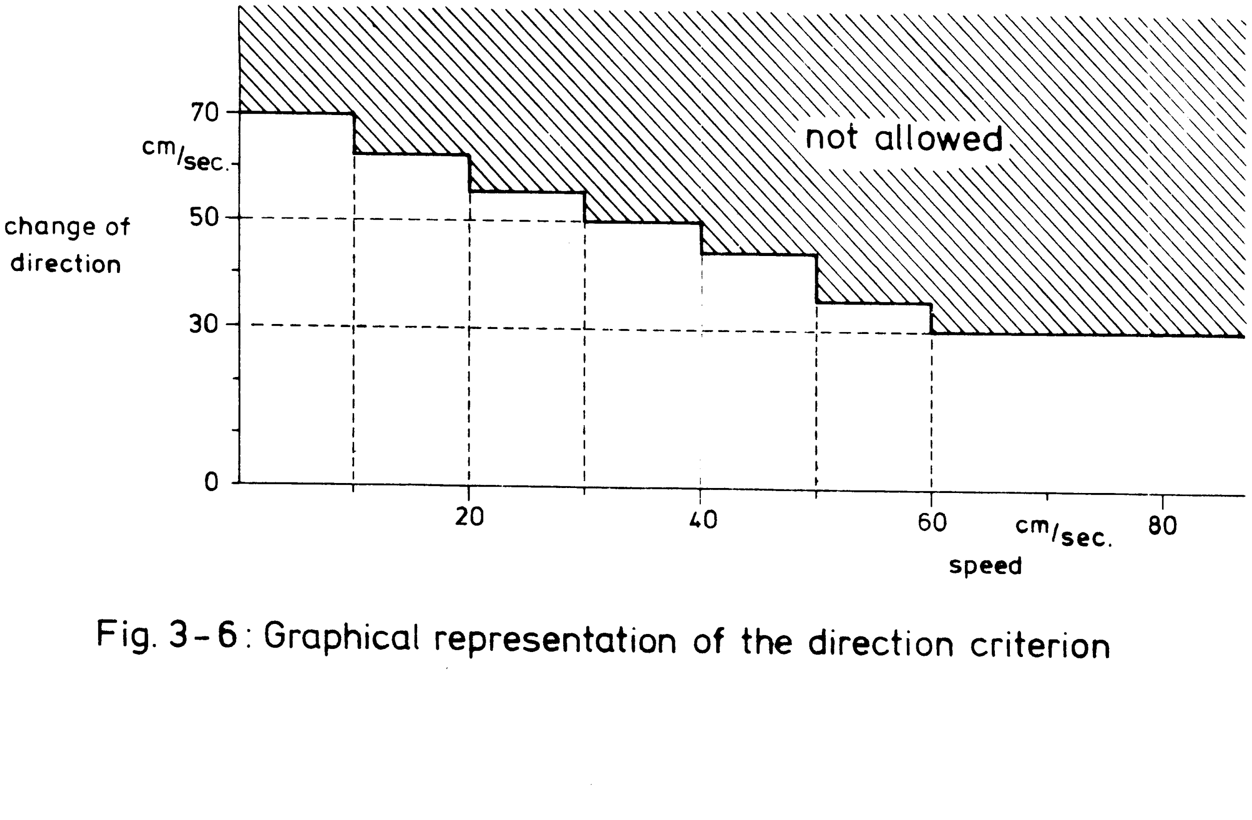
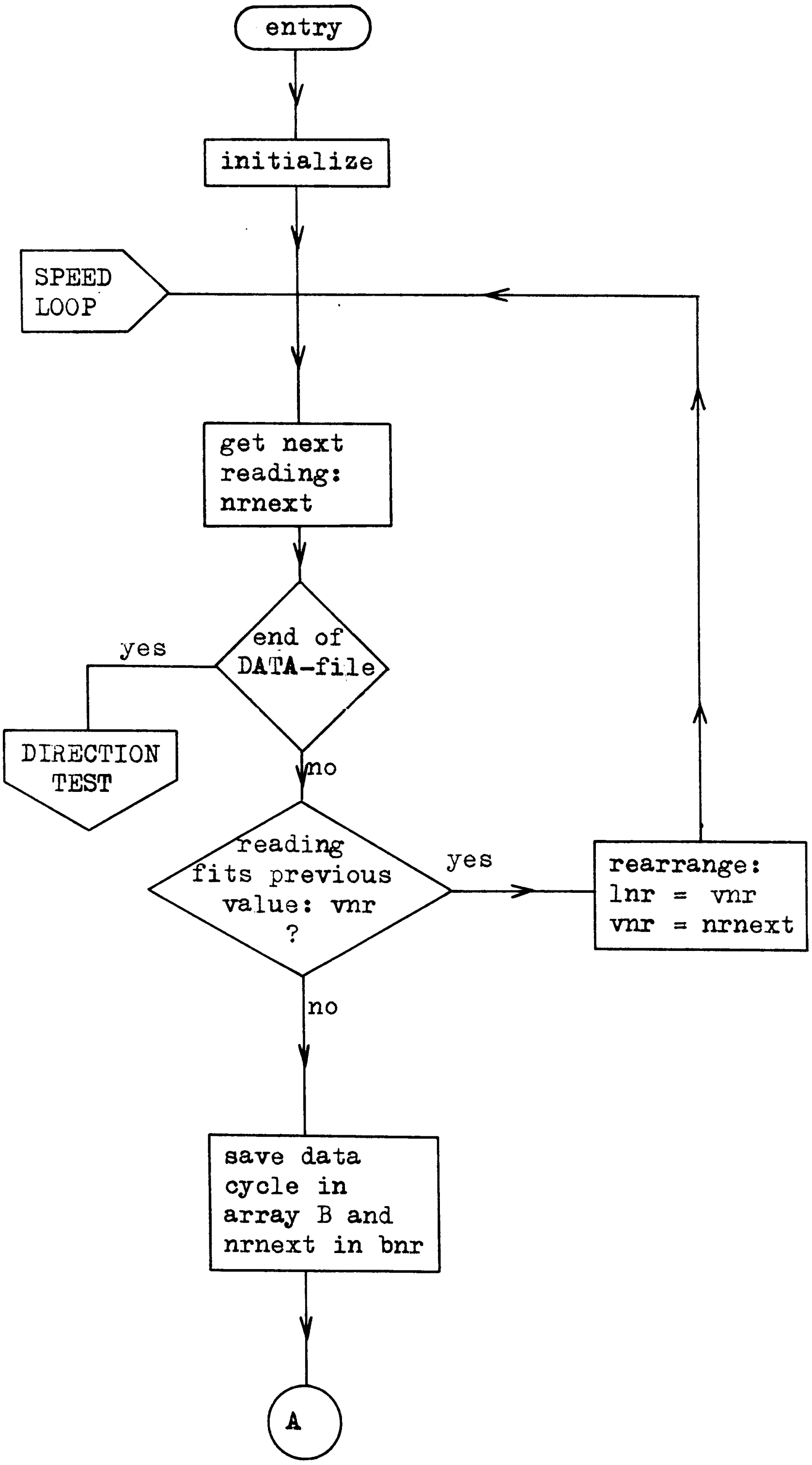


Fig. 3-6: Graphical representation of the direction criterion

Fig. 3-7


entry
initialize
SPEED LOOP
get next reading: nrnext
end of DATA-file
yes
DIRECTION TEST
no
reading fits previous value: vnr ?
yes
rearrange: lnr = vnr vnr = nrnext
no
save data cycle in array B and nrnext in bnr
A

initialize values for "previous value" with number vnr and its predesessor number lnr

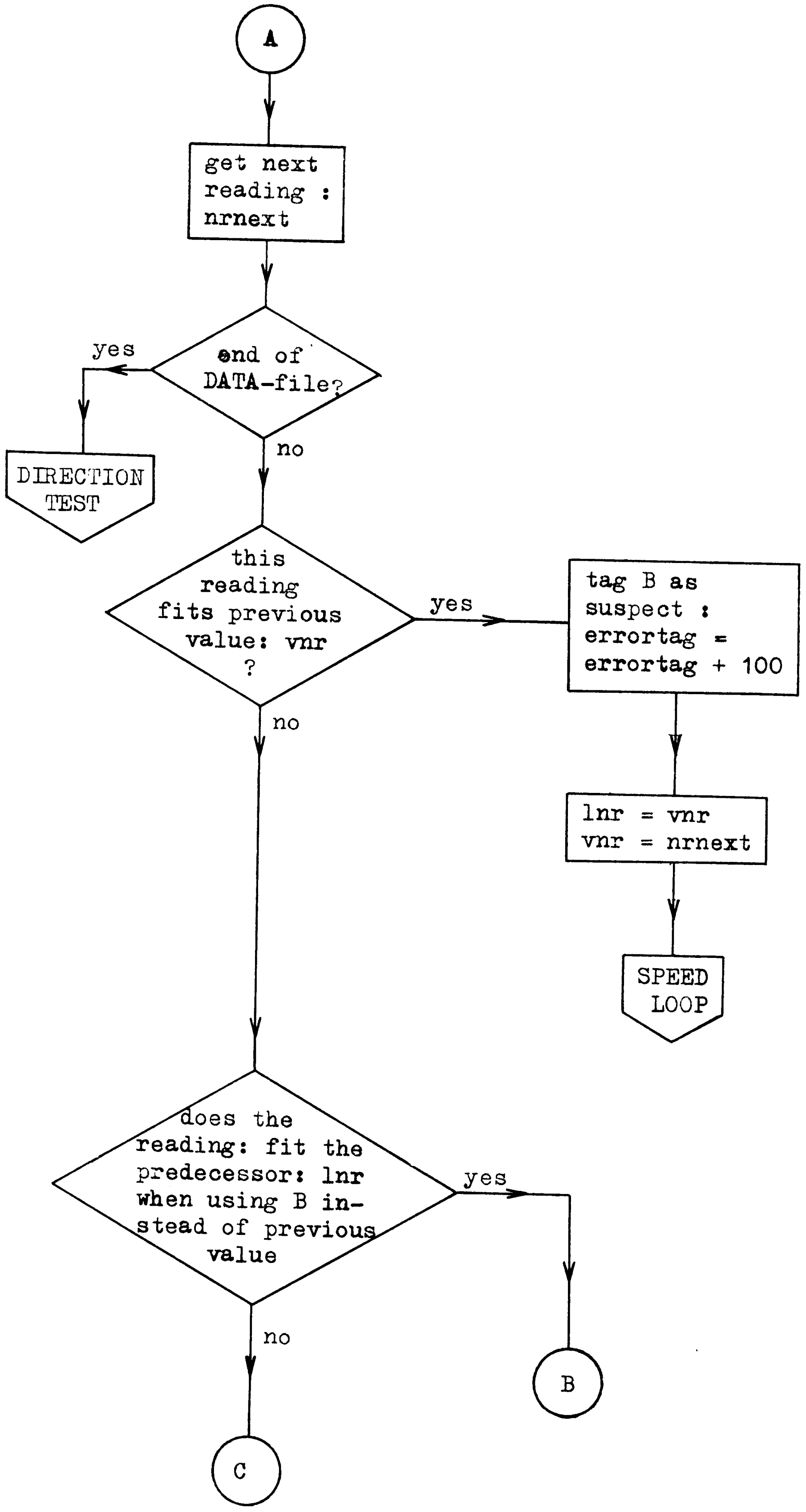
get next reading which is non-gap in the speed channel

if it is the end of DATA-file mark exit from SPEEDLOOP and start direction test

if the present reading fits the previous value rearrange the counters lnr, vnr for future tests.

if the test fails save the full cycle and its number for subsequent use.

Fig. 3-7 continued 1.


A
get next reading : nrnext
end of DATA-file?
yes
DIRECTION TEST
no
this reading fits previous value: vnr ?
yes
tag B as suspect : errortag = errortag + 100
lnr = vnr
vnr = nrnext
SPEED LOOP
no
does the reading: fit the predecessor: lnr when using B in-stead of previous value
yes
B
no
C

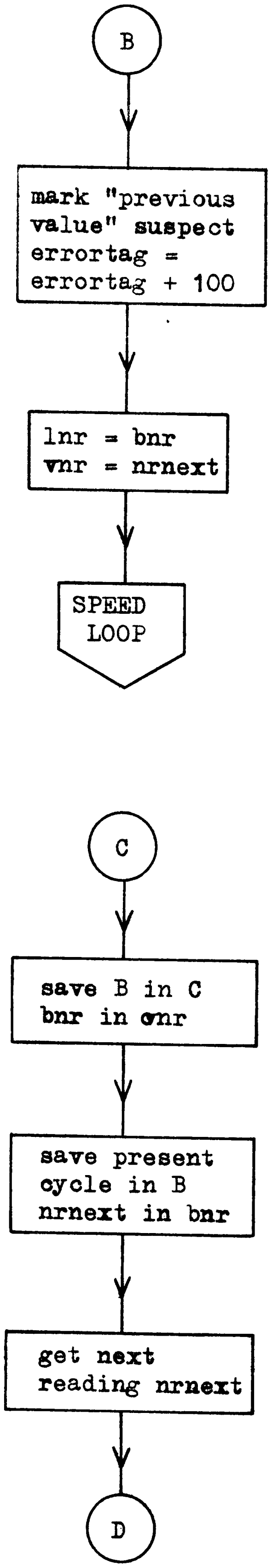
get next non-gap reading

if this reading fits the previous value tag the cycle saved in B as "suspect"

rearrange and re-enter SPEEDLOOP

if it does not, try to fit with the predecessor, lnr using cycle B in stead of the previous value

Fig. 3-7 continued 2.

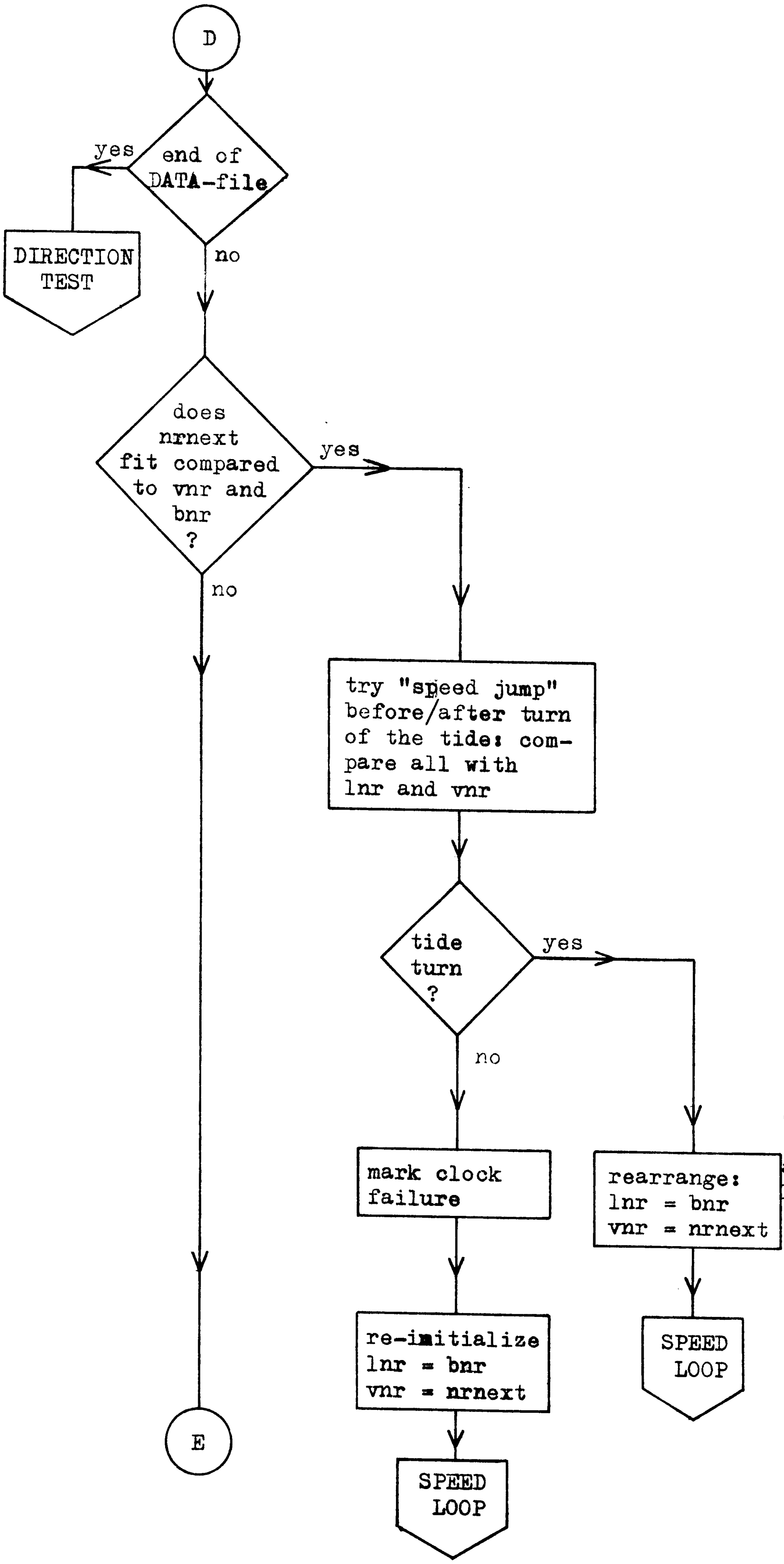


if it does, mark previous value "suspect", rearrange and re-enter SPEEDLOOP

if however the test compared with lnr fails too,
save B and bnr in C and cnr
and save present cycle in B,
its number in bnr

get next non-gap reading

Fig. 3-7 continued 3.


D
end of DATA-file
yes
DIRECTION TEST
no
does nrnext fit compared to vnr and bnr ?
yes
no
try "speed jump" before/after turn of the tide: compare all with lnr and vnr
tide turn ?
yes
no
mark clock failure
re-initialize lnr = bnr vnr = nrnext
SPEED LOOP
rearrange: lnr = bnr vnr = nrnext
SPEED LOOP
E

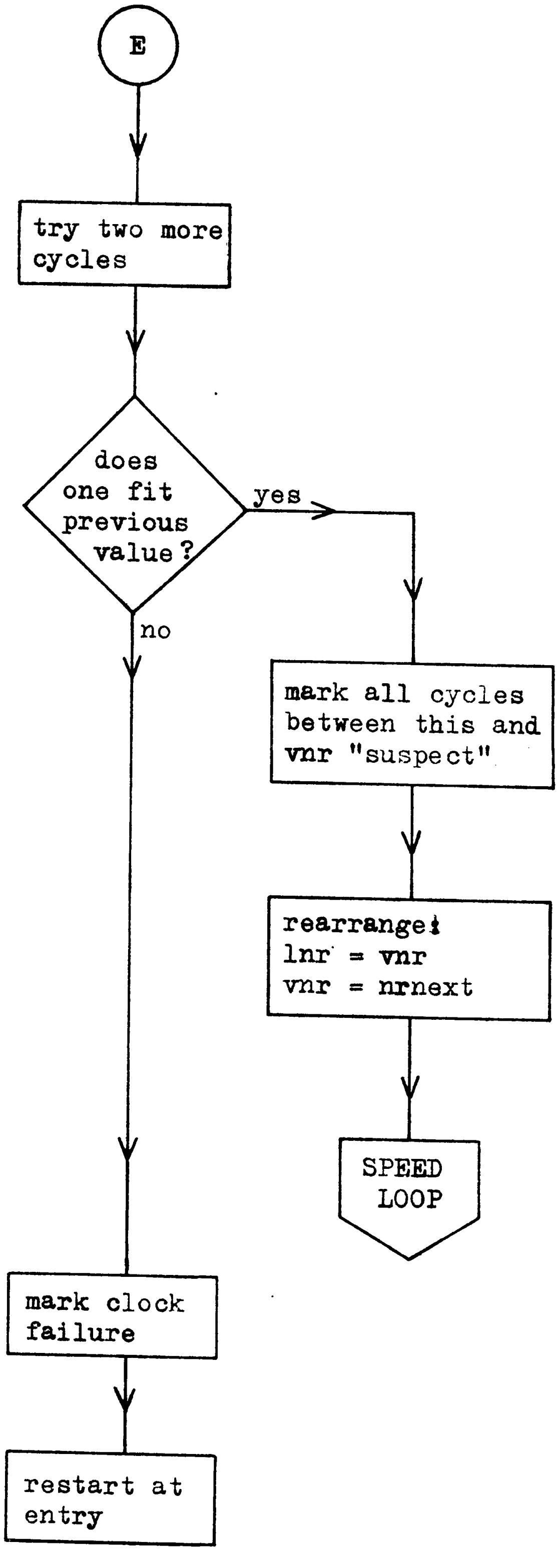
do the three readings in the cycles nrnext, bnr and cnr fit to each other?

If so, look at the time dependence of the readings and compare with lnr and vnr readings.

if it is a "tide turn" rearrange and re-enter SPEEDLOOP

otherwise it is a clock failure which is marked accordingly.
Re-initialize SPEEDLOOP

Fig. 3-7 continued 4.



try to fit the next two cycles to previous value

if one fits mark all cycles in between as "suspect"

rearrange and re-enter SPEEDLOOP

if none fits a clock failure is assumed. Mark accordingly.

and start the full initialization sequence at entry point.

Fig. 3-7 continued 5.

Fig. 3-7 Essentials of the speed channel test flow chart. For full details cf. Appendix D.

The nonsense numbers, as produced by some currentmeters in the gap, and other "suspect" direction values are trapped by a criterion which must properly deal with the following facts:

- direction jumps in alternating tidal currents must be recognized and accepted as such
- at low current speeds the torque which alignes the current meter is small, thereby causing a variability of the direction readings. This variability comprises the combined effects from different sources (mooring motions, influence of surface waves, eddies with time scales of more than a few seconds, etc.) and must be allowed for.

The adopted criterion is again established experimentally and is a function of current speed. At the lowest speed a change of direction from one measurement to the next of about 70 degrees is allowed. This value is reduced stepwise until above 60 cm/sec a change between subsequent measurements of about 30 degrees is allowed.

Since the speed is used as a parameter in the direction criterion it is clear that the direction test cannot be started until the speed channel test came to its end.

Again the criterion in the program has been implemented as a boolean function.

As the test is performed on successive non-gap direction readings provision has been made for intermediate gap-readings, by allowing an extra change of about 15 degrees for each gap-reading encountered between the readings to be compared.

In the test it is assumed that the correlation between two direction readings which are more than about half an hour apart, is so small that testing is no longer considered a sensible action. The reading is then accepted as it is and used in subsequent testing.

With this limitation in mind the test evolves along the following lines:

1. given a "previous direction" get the next non-gap direction and test it with the most recently computed speed.
2. if the test fails get the next direction and test it with an updated speed value as a parameter. If this reading fits, compare it with the rejected value. If this test fails the (twice) rejected value is marked "suspect" in accordance with Table 3-IV. The last accepted reading is taken as the "previous direction" in subsequent tests.
3. If test 2 fails too, get a third non-gap direction reading and compare it with the "previous direction".

   If it fits, the second rejected value is compared with this third value. When rejected it is tagged "suspect". The first rejected reading is then tested with either the second value when this is as yet accepted or with the third reading which is accepted anyhow. If the first direction value is again rejected it is marked "suspect" (cf. Table 3-IV).

Finally the third value is taken as the "previous" value for subsequent test.

4. If the third reading is rejected also, all three values are compared among themselves. Up to three values can then become marked "suspect". As "previous value" we take:
   - the second value when the first and second readings fit but the third does not
   - the third value in all other cases even when marked "suspect". This last possibility reflects the statement that after about half an hour the correlation between direction readings is considered too small to make any sense in using it in tests.

As has already been stressed in section 3.4 for the speed test, the direction test criterion complemented with andwhere necessary corrected by the interpolation procedure for suspect value (cf. section 4.1) is only succesful to the extent that a most critical visual inspection can not give other results.

3.6 Evaluation of clock performance.

As a final test of the full registration, the number of data cycles that must be expected is computed using the input information on the times and dates of the first and the last data-cycles and the time-interval between two successive data-cycles. When this expected number of cycles differs from the decoded number of cycles by less than 2 o/oo a correction for the time-interval is computed and printed. This information combined with the information on detected (rather "suspected") clock failures (cf. section 3.4 on speed channel tests) gives an indication of the quality of registration as far as the timing of the series is concerned. Also a more decisive conclusion about the nature of the "suspected" clock failures can be drawn: if the decoded and the expected number of data cycles are virtually the same a real clock failure is highly improbable if not excluded.

```
        40174 -      43
                                                                                   PAGE        1

 PRAG-DATA-DECODER-MULTIPURPOSE
                                                                            VERSIE:     731003

 K.N.M.I.       DE BILT        NETHERLANDS


      CAMPAIGN          STATION/      INSTRUMENT      WATER      INSTRUMENT     INSTRUMENT
                         PERIOD        DEPTH(M)      DEPTH(M)       TYPE          NUMBER

        7305              202             28            32           22            986

                    DEGR   MIN        DEGR   MIN
 POSITION:           53     24 N        3     0 E


 TIME OF FIRST MEASUREMENT GMT:    19730904      958
 TIME OF  LAST MEASUREMENT GMT:    19731005        0


 TIME INTERVAL IN MINUTES: 10


 OPTIONS USED:

 PROGRAM DECODER                            SUCCESSOR PROGRAM


 TEMPERATURE,                               DATA-COMPUTATION WITH OUTPUT-TAPE
 DIRECTION AND SPEED TEST
```

- identification of the registration and other informative information (type and number of instrument etc.).
- timing information (C[illegible] 11)
- decoding of the function parameter at the end of the first input tape.

```
       40174 -      43
                                                                              PAGE        2
  CHARACTERS BEFORF PEFERENCE:       3

 START TEMP:   620       24
 FOUTE PARITEIT IN METING, KANAAL:     516       3
 FOUTE PARITEIT IN METING, KANAAL:    1639       3
 FOUTE PARITEIT IN METING, KANAAL:    3704       3
 VERTAALFOUT IN METING:     3737
 (NU GOED) ONVOLLEDIGE METING:     3738
 VERTAALFOUT IN METING:     3741
 FOUTE PARITEIT IN METING, KANAAL:    4145       3
```

- 3 non-blank characters found
- start temperature of 620 (binary) units was found in the third hourly mean at datacycle number 24
- parity errors in data cycles 516, 1639 and 3704 in channel number 3 (salinity for an Aanderaa, type = 22)
- translate error in cycle nr. 3737
- registration error in cycle nr. 3738
- translate error in cycle nr. 3741
- parity error in data cycle 4145 in channel 3

```
     40174 -       43
                                                                                                   PAGE       3
19730904
 NR   REF   TEMPERATURE     DIRECTION      VELOCITY     EAST    NORTH
      CH1     CH2          CH3/5 DEGR    CH4/6 CM/S    COMP     COMP          +0       +90      +180      +270      +360                DEGR
  1   986     750            550            210                               .0        .20       .40       .60       .80       .100     CM/S
  2   986     725            122            540
  3   986     725            444            355
  4   986     713            444            770
  5   986     713            444            136
  6   986     742            444            275
  7   986     619            368            335
```

- with the start temp. of 620 the temp. channel is scanned until 3 successive cycles show temperatures with less than 3 binary units difference. The registration is printed until and including the first of those three cycles.
  Data information and cycle numbers are printed with these cycles.

```
     40174 -       43
                                                                                    PAGE       4

TEMP.TEST TIJD (SEC):        10


START STROOMTEST


EERSTE FOUT E GOED

FOUT 2E NDE
FOUT 1FOUT 2    3733     3734      3735
 176        25       111           +81       +86   *   +68
KLOK TEST
SNELH-VERWERKING ONDERBROKEN BIJ         3732

FOUT 1F UT 2FOUT 2FOUT 2SNELHEID-VERWERKING ONDERBROKEN BIJ:    3736

START STROOMTEST

START STROOMTEST

START STROOMTEST


EERSTE DRIE GOED

FOUT 1ALSNOG     EINDESTAR  RICHTINGSTEST      EERSTE GOED
FOUT 1  FOUT 2EINDE B
FOUT 1  FOUT 2EINDE BC
```

1) - in this case the end of the registration was sensed before three successive cycles showed a change of temperature

2) - then the speed test starts

3) - initializing was succesful

4) - in each test route a message is given when the test fails

5) - before a "clock failure" test is done some extra information is printed. In this case the three rejected values so a re-initialization is not needed.

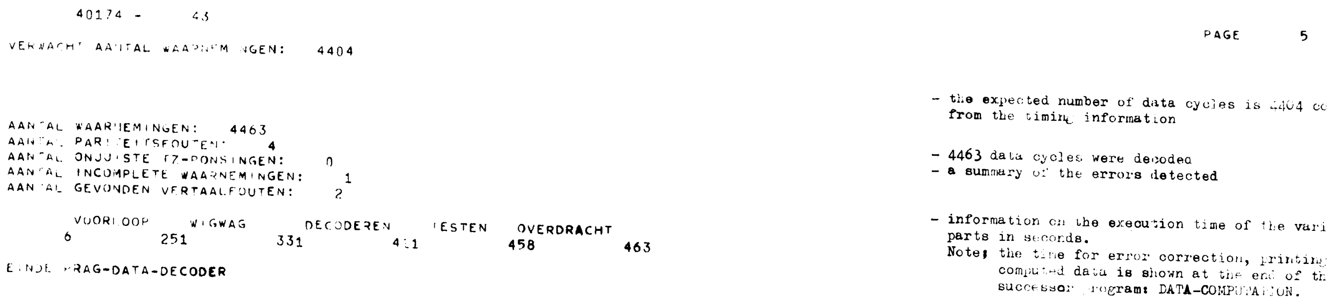
6) - five successive speed data are rejected. A "clock failure" is assumed. Re-initialization is necessary.

7) - This re-initialization must be retried twice before being successful

8) - start of direction test

9) - first value accepted

10) - as in the speed test, each path gives an output when the test fails,



```
        40174 -      43                                                                        PAGE      5

VERWACHT AANTAL WAARNEMINGEN:    4404



AANTAL WAARNEMINGEN:    4463
AANTAL PARITEITSFOUTEN:     4
AANTAL ONJUISTE FZ-PONSINGEN:       0
AANTAL INCOMPLETE WAARNEMINGEN:       1
AANTAL GEVONDEN VERTAALFOUTEN:      2

         VOORLOOP      WIGWAG        DECODEREN      TESTEN     OVERDRACHT
       6          251          331          411          458          463

EINDE KRAG-DATA-DECODER
```

- the expected number of data cycles is 4404 computed from the timing information

- 4463 data cycles were decoded
- a summary of the errors detected

- information on the execution time of the various parts in seconds.
  Note; the time for error correction, printing the computed data is shown at the end of the successor program: DATA-COMPUTATION.

Fig. 3-8. Typical output DATA-DECODER-MULTIPURPOSE.

3.7. Explanation of the output.

In fig. 3-8 an output of the DECODER-program is shown along with comments. Moreover it is not an output of the program version as given in Appendix D in which same minor program errors with no bearing on the output have been eliminated.

Also not all possible errors occurred in this registration.

3.8. Summary of results and timing information.

Compared to the results of the older programs, described in Report no. V-241 of the K.N.M.I. (ref. 1), the following improvements have been achieved:

- reduction of the execution time for the complete decode and test sequence by a factor of 4-5.
- proper indication of erroneous registration cycles by using the structure definition of Table 3-III.
- more detailed quality control of the translation process by introduction of the translate error notion and by inspection of the true zero bits.
- test criteria independent of the tidal characteristics of the measuring area have been devised.
- introduction of a clock failure test has revealed a number of peculiarities of the (mechanical) clocks in Plessey and Aanderaa current meters. This gives us a better indication of the quality of the registration.
- the tests give the same results as a most critical visual inspection could give. Therefore such visual inspection can be greatly reduced and are guided by appropriate messages from the program. Also the servicing of the instruments is facilitated by these messages.
- the program is arranged in such way that extensions and/or changes can be incorporated rather easily.

As far as computer time is concerned it can be stated that on the Electrologica EL-X8 computer installation in our Institute the decode and test sequence together with the initial read in of input tapes for a typical 6 week registration takes about 6 minutes job run time. In a multiprogramming environment the CPU time could probably be reduced to about 60-70 % of that time.

4. DATA-COMPUTATION.

The program DATA-COMPUTATION embodies the second job step when one of the functions 0 through 3 has been asked for (cf. Table 3-II).

The main purpose of DATA-COMPUTATION is the interpolation of "gap" and "suspect" direction and velocity readings under control of the errortags in the DATA-file and the listing of the results computed from the (thus updated) DATA-file. Besides hourly means of a number of channels are computed and listed at the end of the printout.

The interpolation method is developed and discussed in section 4.1. In section 4.2 various pages of the output are shown and commented. An example of what can be done by visual inspection of the output is discussed in section 4.3.

In section 4.4 some results are summarized along with information on job run time.

4.1. Handling of detected errors.

In chapter 3 three different kinds of errors have been introduced (cf. Table 3-IV):

a. incomplete data cycles with errortag = 1

b. "gap" readings in the direction or velocity channels or both, indicated by errortag = 10, 100 or 110

c. "suspect" readings in the direction or velocity channels or both, again marked by errortag = 10, 100 or 110.

In the case of an incomplete cycle the direction and velocity channel readings must be synthesized by interpolation. The same applies for "gap" readings. "Suspect" readings, however, require a more elaborate treatment, because they are suspected to be errors but might as well be sensible readings, which could not fit the test criteria for one or more reasons. Among these reasons are:

- stormy weather (more accurately: a rough sea-surface with seas or swells) can cause a noisy signal, especially in the direction channel
- in an area with alternating tidal currents, the period around the turn of the tide can yield data for direction and/or velocity that individually still cannot be made to fit the test criteria although it is found to fit the overall time dependence.

We therefore developed a scheme in which the difference between the "suspect" reading and the value resulting from the interpolation is compared with the scatter of the correct readings around a computed least-squares fit. If that difference is less than twice the r.m.s. error of the fit, the reading is accepted after all and the errortag updated accordingly. Otherwise the "suspect" value is indeed rejected and the interpolated value will be used instead.

The interpolations of the direction and speed channels is done in separate passes.

For the least-squares fit computation an algorithm is used which selects as a minimum 5 non-gap, non-suspect values in the channel concerned preceding a gap or suspect value and 5 accepted data following the value to be interpolated. If additional gap or suspect values occur in these groups of 5 the length of the "interpolation area" is accordingly increased until 5 successive accepted values are found at the beginning and at the end of the area. This requirement might give problems at the begin and at the end of the total time series and is therefore slightly modified to fit the boundaries of the time series.

The data thus selected are fed to a subroutine which computes the least-squares fit polynomial. The degree of this polynomial is selected by the subroutine to give the best fit but has an upper limit given as a parameter.

In our program we have chosen this upper limit to be 5 as an empirically acceptable value. Using a minimum of 10 data points the maximum of 6 coefficients is found to give reasonable interpolations, even for directions at and around the tide-turn in purely alternating currents.

After the computation of the least-squares fit and the r.m.s. error of the fit as compared to the data points on which the fit is based, the (nonsense) data in incomplete cycles and gap-readings in the pertaining channel inside the "interpolation area" considered are replaced by values computed with the least-squares fit polynomial.

"Suspect" readings are tested to be in a range of twice the r.m.s. error on either side of the appropriate polynomial value.

As already mentioned above the errortag is updated if the "suspect" reading is within that range and is accordingly accepted. If the test fails, the "suspect" reading is replaced by the polynomial value. Both numbers -suspect and corrected value- are listed for visual inspection afterwards

This procedure clearly corrects for shortcomings of the test criteria for speed and direction, especially under such circumstances as large trends (at tide-turn e.g.) or noisy signals (bad weather or swells). Therefore we consider this correction procedure as essential in order to have the rather static testcriteria look a little more dynamic, thus resulting in a setup which performs very well under widely different conditions.

4.2. Explanation of the output.

The program DATA-COMPUTATION expects two different blocks of data:

- identification and calibration data as written on backing (drum) storage by the DATA-DECODER program
- the DATA-file also on secondary storage. The program DATA-COMPUTATION starts by reading and listing the identification and calibration data (Fig. 4-1).

Then the DATA-file is scanned for errortags corresponding to the speed channel. If such an errortag is encountered having the value 1, 100 or 110 (cf. Table 3-IV) the number of the pertaining data cycle is entered in an errortable and listed at the same time (Fig. 4-2), if any.

This scan is continued until the end of the DATA-file or a "clock failure" mark is sensed.

Under control of the errortable the interpolations are done. The replacements done are listed in a table, as shown in Fig. 4-3.

The same sequence (scanning the errortag and interpolation) is performed for the direction channel, also for that part of the DATA-file until the end of file or "clock failure" mark is encountered, whichever comes first (see Figs. 4-4 and 4-5).

Then the individual data cycles are converted to engineering units and printed as shown in Fig. 4-6.

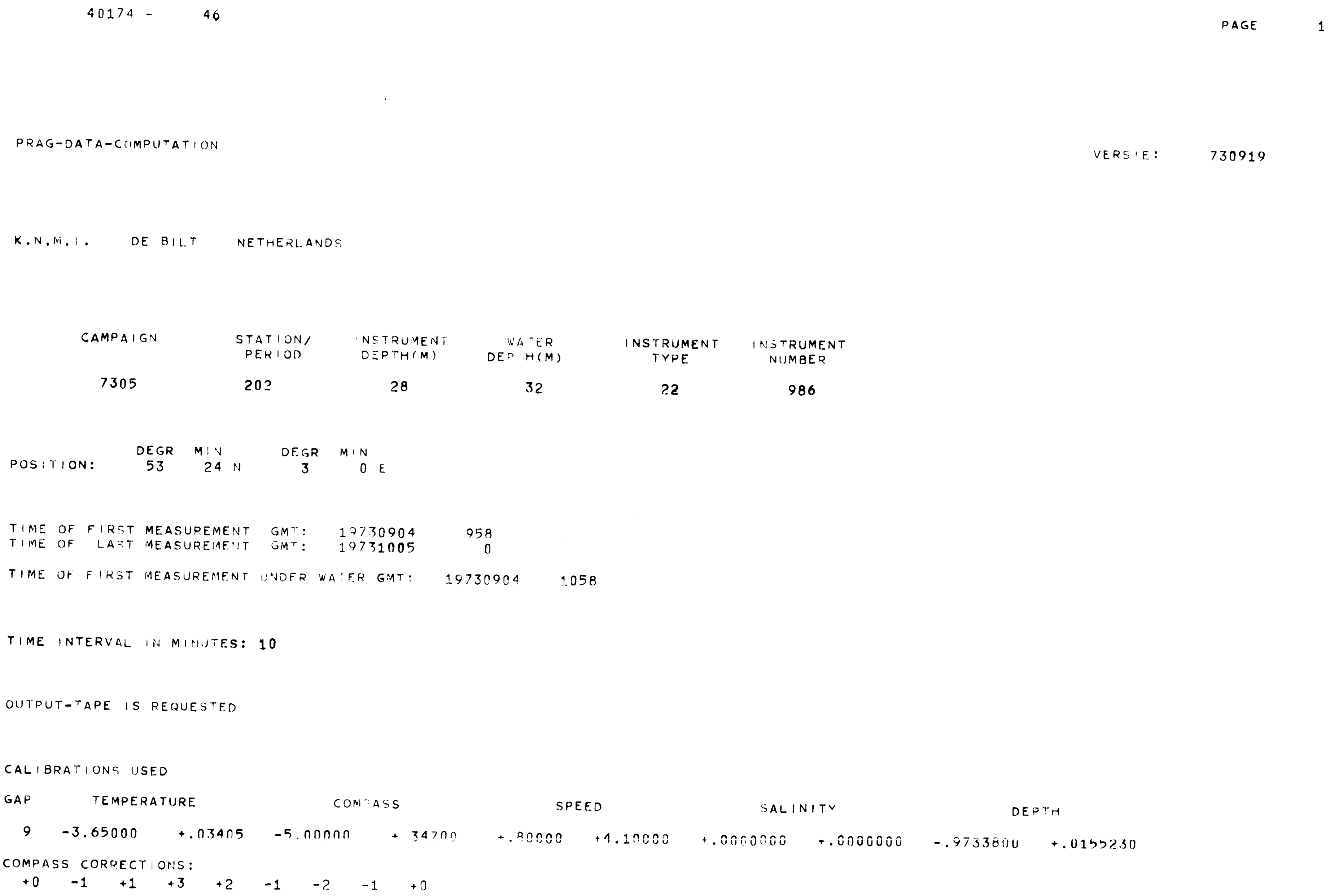
Since the speed shown is in fact an average over the (10 minute) time interval, the direction in degrees is computed as an average of the converted reading in the corresponding data cycle and the preceding converted direction reading.

Values that have been interpolated are marked by V (velocity channel) or D (direction channel) whereas interpolated, incomplete cycles are marked by an A between the table and the graphical representation of direction (+) and speed (.).

Also data and (approximate) time information is given in this listing.

Following this listing, computed hourly means for those quantities that have been measured are given, Fig. 4-7. The averaging is done in a straightforward manner, by summing the data and subsequent division.

Finally counting and timing information is given, see Fig. 4-8^a. The whole sequence of errorchecking, interpolation, data listing, hourly means table and run time statistics is repeated for each part of the DATA-file terminated by a clock failure or the final end of file mark.



```
    40174 -    46                                                                                              PAGE      1



PRAG-DATA-COMPUTATION                                                                              VERSIE:    730919



K.N.M.I.    DE BILT    NETHERLANDS



     CAMPAIGN          STATION/     INSTRUMENT     WATER        INSTRUMENT    INSTRUMENT
                        PERIOD       DEPTH(M)     DEPTH(M)        TYPE          NUMBER

       7305              202           28            32            22            986



             DEGR   MIN      DEGR   MIN
POSITION:     53    24 N       3     0 E



TIME OF FIRST MEASUREMENT  GMT:   19730904     958
TIME OF  LAST MEASUREMENT  GMT:   19731005       0

TIME OF FIRST MEASUREMENT UNDER WATER GMT:   19730904    1058



TIME INTERVAL IN MINUTES: 10



OUTPUT-TAPE IS REQUESTED



CALIBRATIONS USED

GAP      TEMPERATURE                 COMPASS                SPEED                  SALINITY                 DEPTH

 9   -3.65000   +.03405   -5.00000   + 34700   +.80000   +4.10000   +.0000000   +.0000000   -.9733800   +.0155230

COMPASS CORRECTIONS:
 +0   -1   +1   +3   +2   -1   -2   -1   +0
```

Fig. 4-1. Listing of identification and calibration data.

```
     40174 -      46                                                                                          PAGE      2

 FOUTENTABEL SNELHEID
     86       129       221       252       300       326       407       509       543
    570       794       809       828       663       888       935       996      1056      1069
   1269      1347      1385      1397      1412      1468      1489      1513      1620      1760
   1826      1840      1869      1992      2188      2210      2223      2341      2367      2575
   2759      2800      2889      2938      3023      3051      3069      3264      3359      3374
   3475      3488      3501      3637      3707
```

Fig. 4-2. Listing of the contents of the errortable for the speed channel.

```
     40174 -      46                                                                                  PAGE       3

SPEED INTERPOLATIONS AT:
    NR     OLD    NEW   SORT
     86     +0     +2   V
    129     +1     +9   V
    221  +1023  +1027   V
    252     +0     +4   V
    300     +1     +2   V
    326  +1023  +1023   V
    407     +0     +1   V
    509  +1023  +1026   V
    543  +1023  +1026   V
    570  +1023  +1031   V
    794     +0     +3   V
    809  +1023  +1016   V
    828  +1023  +1026   V
    863  +1023  +1031   V
    888  +1023  +1032   V
    935  +1023  +1028   V
    996  +1023  +1029   V
   1056  +1023  +1025   V
   1069  +1023  +1027   V
   1269  +1023  +1025   V
   1347  +1023  +1026   V
   1385  +1023  +1024   V
   1397  +1008  +1031   V
   1412     +0     +1   V
   1468  +1023  +1026   V
   1489  +1023  +1024   V
   1513  +1023     +5   V
   1620  +1023     +2   V
   1760  +1023  +1028   V
   1826  +1023  +1031   V
   1840  +1023  +1028   V
   1869  +1023  +1026   V
   1992  +1023  +1023   V
   2188  +1023  +1029   V
   2210  +1023  +1032   V
   2223  +1023  +1021   V
   2341  +1023  +1022   V
   2367  +1023  +1030   V
   2575  +1023  +1031   V
   2759  +1023  +1031   V
   2800  +1023  +1026   V
   2889  +1023  +1023   V
   2938  +1023  +1028   V
   3023  +1023  +1025   V
   3051     +0  +1032   V
   3069  +1023  +1027   V
   3264     +0     +4   V
   3359     +1     +4   V
   3374  +1023     +1   V
   3475  +1023  +1023   V
   3488  +1023  +1019   V
   3501  +1023  +1025   V
   3637  +1023  +1020   V
   3707  +1023  +1032   V
```

Speed interpolations can have two "SORT"s':

V gap or "suspect" (and rejected) value
A incomplete cycle; no "OLD" value will then be printed.

In this case only cycle nr. 1397 is rejected; all other values are gap-readings.

Fig. 4-3. Listing of interpolated speed values.

```
      40174 -      46                                                                                      PAGE     4

FOUTENTABEL RICHTING
      51        55       130       131       132       203       204       205       206
     353       354       355       356       437       438       439       440       512       513
     515       585       588       661       663       664       813       886       887       957
     960      1030      1031      1032      1039      1105      1106      1180      1181      1401
    1402      1472      1475      1478      1547      1548      1619      1620      1625      1694
    1695      1696      1765      1766      1840      1841      1914      1915      1916      1989
    2062      2063      2135      2138      2210      2213      2285      2286      2288      2294
    2366      2445      2446      2448      2449      2526      2595      2670      2746      2747
    2748      2822      2894      2895      2896      2898      2972      2973      2974      3119
    3122      3192      3193      3265      3337      3340      3344      3408      3411      3414
    3482      3487      3489      3558      3560      3561      3632      3636      3699      3702
```

Fig. 4-4. Listing of direction error table.

```
     40174 -       46                                                                    PAGE      5

DIRECTION INTERPOLATIONS AT:
   NR     OLD    NEW   SORT
     51  +1023  +1016   D
     55     +0     +5   D
    130  +1023  +1033   D
    131     +0     +8   D
    132  +1023    +16   D
    203  +1023  +1007   D
    204  +1023  +1008   D
    205     +0  +1013   D
    206  +1023  +1021   D
    353  +1023  +1022   D
    354  +1023  +1026   D
    355  +1023  +1032   D
    356  +1023     +1   D
    437  +1023  +1017   D
    438     +1  +1022   D
    439     +0  +1028   D
    440     +1  +1035   D
    512  +1023  +1036   D
    513     +1     +6   D
    515     +0    +15   D
    585  +1023  +1009   D
    588     +0  +1018   D
    661  +1023  +1012   D
    663  +1023  +1034   D
    664     +1     +6   D
    813  +1023  +1013   D
    886  +1023  +1030   D
    887  +1023  +1037   D
    957  +1023  +1005   D
    960  +1023    +11   D
   1030     +0  +1026   D
   1031  +1023  +1034   D
   1032     +0     +3   D
   1039     +0    +28   D
   1105     +1  +1014   D
   1106  +1023  +1022   D
   1180  +1023  +1029   D
   1181  +1023  +1036   D
   1401  +1023  +1028   D
   1402  +1023  +1035   D
   1472  +1023  +1033   D
   1475     +1     +9   D
   1478     +1    +19   D
   1547  +1023  +1026   D
   1548  +1023  +1033   D
   1619  +1023  +1020   D
   1620     +1  +1026   D
   1625  +1023    +12   D
   1694  +1023  +1021   D
   1695  +1023  +1026   D
   1696  +1023  +1033   D
   1765     +0  +1017   D
   1766     +0  +1029   D
   1840  +1023  +1028   D
   1841  +1023     +2   D
   1914  +1023  +1023   D
   1915     +0  +1036   D
   1916  +1023    +12   D
```

Direction interpolations can be either of two "SORT"s:

D gap or rejected values

A incomplete cycle; no "OLD" value will be printed.

Fig. 4-5. Table of direction interpolations (cont).

```
     40174 -      46                                                                PAGE      9

   19730905
    NR    REF    TEMPERATURE       DIRECTION        VELOCITY        EAST    NORTH
          CH1     CH2             CH3/5 DEGR       CH4/6 CM/S       COMP     COMP

      1   986     621 +17.5        413   147        2 V   22.1     +11.9    -18.6
      2   986     621 +17.5        454   153       55     22.5     +10.2    -20.1
      3   986     621 +17.5        454   160      112     24.2      +8.3    -22.7
      4   986     622 +17.5        458   161      167     23.3      +7.8    -22.0
      5   986     622 +17.5        468   163      222     23.3      +6.8    -22.3
      6   986     622 +17.5        471   165      280     24.6      +6.3    -23.8
   HOURLY MEAN  1
      7   986     622 +17.5        461   164      337     24.2      +6.7    -23.2
      8   986     622 +17.5        476   165      394     24.2      +6.3    -23.3
      9   986     622 +17.5        492   170      455     25.8      +4.4    -25.4
     10   986     622 +17.5        503   175      511     23.8      +2.2    -23.7
     11   986     622 +17.5        489   174      566     23.3      +2.4    -23.2
     12   986     622 +17.5        515   176      619     22.5      +1.5    -22.5
   HOURLY MEAN  2
     13   986     622 +17.5        544   185      670     21.7      -2.0    -21.6
     14   986     622 +17.5        548   190      719     20.9      -3.8    -20.5
     15   986     623 +17.6        592   198      767     20.5      -6.4    -19.4
     16   986     623 +17.6        604   207      808     17.6      -8.1    -15.6
     17   986     623 +17.6        627   213      848     17.2      -9.4    -14.4
     18   986     623 +17.6        625   216      889     17.6     -10.5    -14.2
   HOURLY MEAN  3
     19   986     623 +17.6        658   222      930     17.6     -11.7    -13.2
     20   986     623 +17.6        667   229      972     18.0     -13.5    -11.9
     21   986     623 +17.6        680   232     1013     17.6     -14.0    -10.7
     22   986     623 +17.6        692   237       17     15.6     -13.0     -8.6
     23   986     623 +17.6        708   241       57     17.2     -15.1     -8.2
     24   986     623 +17.6        717   246       93     15.6     -14.2     -6.4
   HOURLY MEAN  4
     25   986     623 +17.6        731   250      130     16.0     -15.0     -5.6
     26   986     623 +17.6        751   255      170     17.2     -16.6     -4.4
     27   986     623 +17.6        773   262      212     18.0     -17.9     -2.4
     28   986     623 +17.6        776   267      251     16.8     -16.8      -.9
     29   986     622 +17.5        792   270      294     18.4     -18.4      +.1
     30   986     622 +17.5        790   273      336     18.0     -18.0      +.8
   HOURLY MEAN  5
     31   986     622 +17.5        825   279      381     19.2     -19.0     +2.9
     32   986     622 +17.5        836   287      427     19.7     -18.8     +5.6
     33   986     622 +17.5        865   294      470     18.4     -16.9     +7.4
     34   986     622 +17.5        873   300      517_    20.1     -17.3    +10.1
     35   986     622 +17.5        871   301      566     20.9     -17.8    +10.9
     36   986     622 +17.5        912   308      613     20.1     -15.7    +12.4
   HOURLY MEAN  6
     37   986     622 +17.5        921   317      663     21.3     -14.5    +15.6
     38   986     622 +17.5        925   320      716     22.5     -14.6    +17.1
     39   986     622 +17.5        946   324      771     23.3     -13.7    +18.9
     40   986     622 +17.5        973   332      828     24.2     -11.2    +21.4
     41   986     622 +17.5        962   335      881     22.5      -9.4    +20.5
     42   986     622 +17.5       1001   340      933     22.1      -7.5    +20.8
   HOURLY MEAN  7
     43   986     622 +17.5        987   345      987     22.9      -6.1    +22.1
     44   986     622 +17.5       1016   347        9 V   22.9      -5.0    +22.4
     45   986     621 +17.5       1033 D 355       62     22.5      -1.8    +22.5
     46   986     621 +17.5          8 D   0      123     25.8       +.2    +25.8
     47   986     621 +17.5         16 D   4      175     22.1      +1.5    +22.1
     48   986     621 +17.5         29     8      225     21.3      +2.8    +21.1
   HOURLY MEAN  8
```

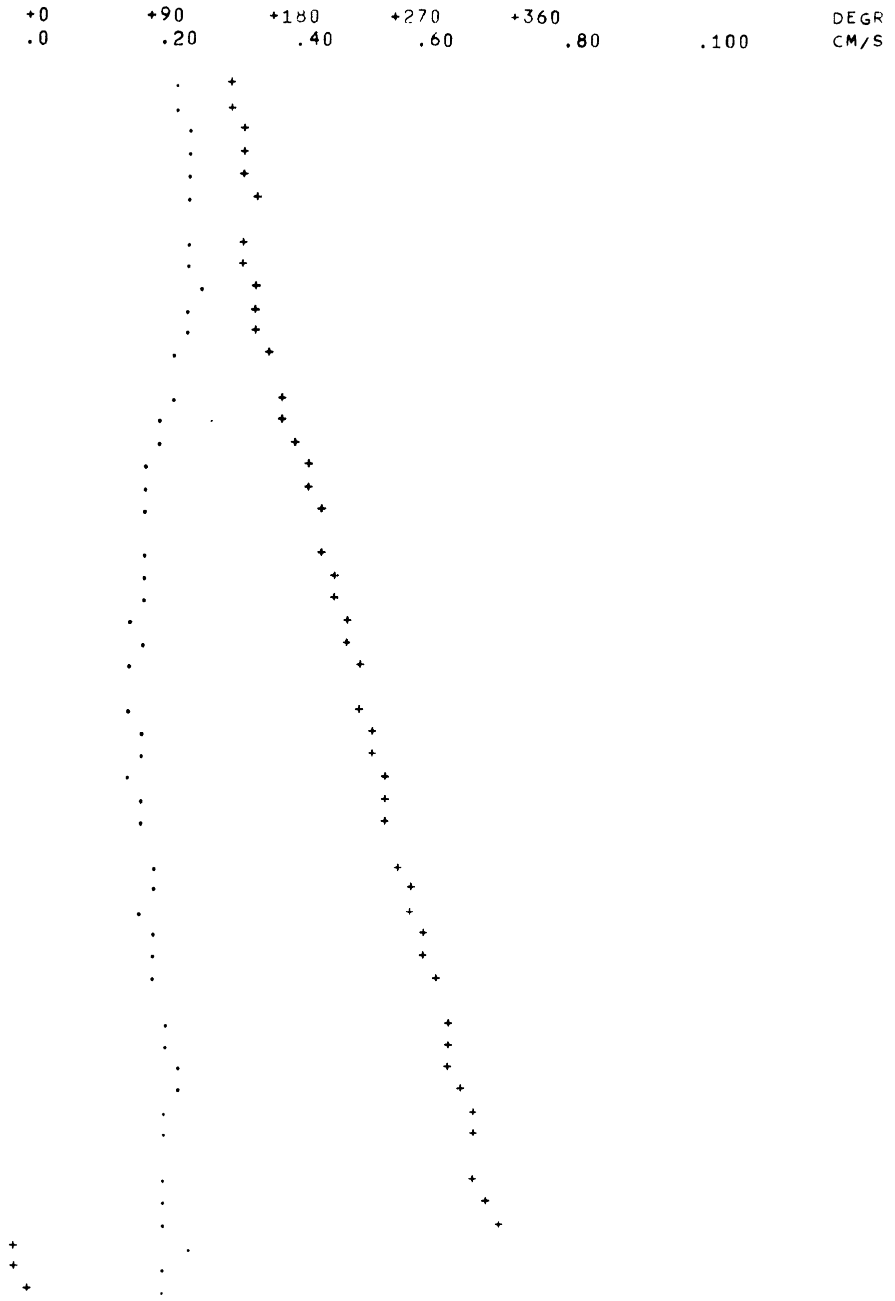
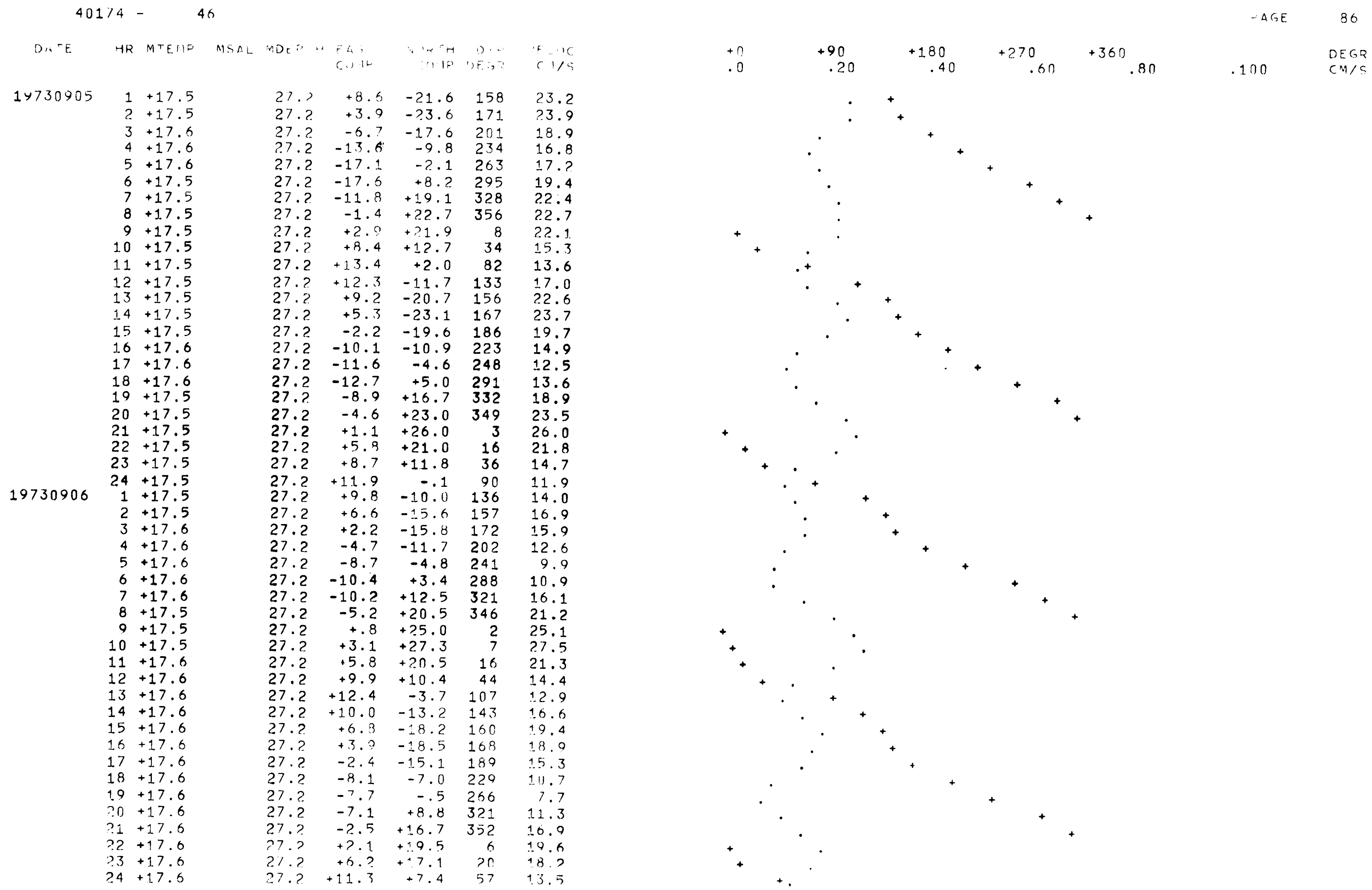


Fig. 4-6. Part of the printout of converted data cycles.



```
     40174 -      46                                                       PAGE      86

  DATE      HR MTEMP   MSAL  MDEPTH  EAST    NORTH   DIR    VELOC
                                     COMP    COMP    DEGR   CM/S

19730905     1 +17.5          27.2    +8.6   -21.6   158    23.2
             2 +17.5          27.2    +3.9   -23.6   171    23.9
             3 +17.6          27.2    -6.7   -17.6   201    18.9
             4 +17.6          27.2   -13.6    -9.8   234    16.8
             5 +17.6          27.2   -17.1    -2.1   263    17.2
             6 +17.5          27.2   -17.6    +8.2   295    19.4
             7 +17.5          27.2   -11.8   +19.1   328    22.4
             8 +17.5          27.2    -1.4   +22.7   356    22.7
             9 +17.5          27.2    +2.9   +21.9     8    22.1
            10 +17.5          27.2    +8.4   +12.7    34    15.3
            11 +17.5          27.2   +13.4    +2.0    82    13.6
            12 +17.5          27.2   +12.3   -11.7   133    17.0
            13 +17.5          27.2    +9.2   -20.7   156    22.6
            14 +17.5          27.2    +5.3   -23.1   167    23.7
            15 +17.5          27.2    -2.2   -19.6   186    19.7
            16 +17.6          27.2   -10.1   -10.9   223    14.9
            17 +17.6          27.2   -11.6    -4.6   248    12.5
            18 +17.6          27.2   -12.7    +5.0   291    13.6
            19 +17.5          27.2    -8.9   +16.7   332    18.9
            20 +17.5          27.2    -4.6   +23.0   349    23.5
            21 +17.5          27.2    +1.1   +26.0     3    26.0
            22 +17.5          27.2    +5.8   +21.0    16    21.8
            23 +17.5          27.2    +8.7   +11.8    36    14.7
            24 +17.5          27.2   +11.9     -.1    90    11.9
19730906     1 +17.5          27.2    +9.8   -10.0   136    14.0
             2 +17.5          27.2    +6.6   -15.6   157    16.9
             3 +17.6          27.2    +2.2   -15.8   172    15.9
             4 +17.6          27.2    -4.7   -11.7   202    12.6
             5 +17.6          27.2    -8.7    -4.8   241     9.9
             6 +17.6          27.2   -10.4    +3.4   288    10.9
             7 +17.6          27.2   -10.2   +12.5   321    16.1
             8 +17.5          27.2    -5.2   +20.5   346    21.2
             9 +17.5          27.2     +.8   +25.0     2    25.1
            10 +17.5          27.2    +3.1   +27.3     7    27.5
            11 +17.6          27.2    +5.8   +20.5    16    21.3
            12 +17.6          27.2    +9.9   +10.4    44    14.4
            13 +17.6          27.2   +12.4    -3.7   107    12.9
            14 +17.6          27.2   +10.0   -13.2   143    16.6
            15 +17.6          27.2    +6.3   -18.2   160    19.4
            16 +17.6          27.2    +3.9   -18.5   168    18.9
            17 +17.6          27.2    -2.4   -15.1   189    15.3
            18 +17.6          27.2    -8.1    -7.0   229    10.7
            19 +17.6          27.2    -7.7     -.5   266     7.7
            20 +17.6          27.2    -7.1    +8.8   321    11.3
            21 +17.6          27.2    -2.5   +16.7   352    16.9
            22 +17.6          27.2    +2.1   +19.5     6    19.6
            23 +17.6          27.2    +6.2   +17.1    20    18.2
            24 +17.6          27.2   +11.3    +7.4    57    13.5
```

Fig. 4-7. Example of hourly means table.

```
     40174 -       46                                                                                                       PAGE      99

AANTAL WAARNEMINGEN:    4463
AANTAL GEPONSTE WAARNEMINGEN:    3725

      OVERDRACHT  ERRORFILL    CORRIGEREN    REKENEN     UURTABEL
     5             7            89          154           644           712
```

Total number of data cycles
number of data cycles punched sofar

Fig. 4-8a. Counting and timing information for part of the DATA-file until first clock failure.

```
        40174 -      46                                                                                               PAGE     106

     AANTAL WAARNEMINGEN:    4463
     AANTAL GEPONSTE WAARNEMINGEN:    3729

        OVERDRACHT   ERRORFILL   CORRIGEREN   REKENEN    UURTABEL
     712          712          712          714          717          718
```

Fig. 4-8^b. As Fig. 4-8^a but for part between first and second clock failure.

```
         40174 -      46                                                                                                                              PAGE    132

   AANTAL WAARNEMINGEN:    4463
   AANTAL GEPONSTE WAARNEM NGEN:    4456

         OVERDRACHT  ERRORFILL   CORRIGEREN   REKENEN    UURTABEL
      719          719         736          750          846          860

   EINDE PRAG-DATA -COMPUTATION
```

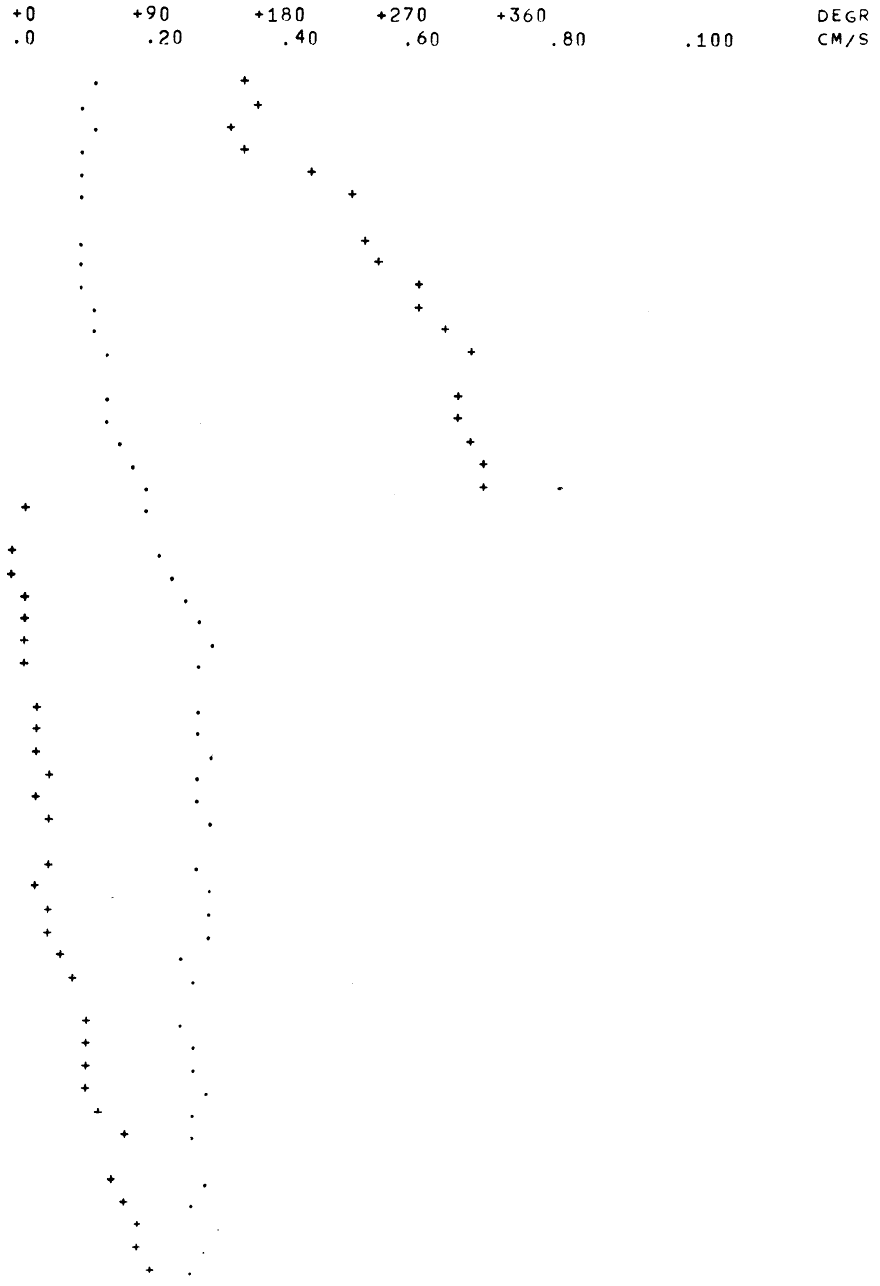
Fig. 4-8^c. As Fig. 4-8^a but for DATA-file part between second clock failure and end-of-file.

```
1973093)
 NR    REF   TEMPERATURE       DIRECTION      VELOCITY      EAST   NORTH
       CH1    CH2             CH3/5 DEGR     CH4/6 CM/S     COMP    COMP

  1    986    585 +16.3        541   173      277    12.3    +1.5   -12.2
  2    986    585 +16.3        480   179      300    10.2     +.2   -10.2
  3    986    584 +16.2        416   158    " 326    11.5    +4.3   -10.6
  4    986    585 +16.3        580   174      349    10.2    +1.0   -10.2
  5    986    584 +16.2        664   215      373    10.6    -6.2    -8.7
  6    986    584 +16.2        805   253      397    10.6   -10.2    -3.0
HOURLY MEAN  1
  7    986    584 +16.2        683   257      418     9.4    -9.2    -2.2
  8    986    584 +16.2        896   273      441    10.2   -10.2     +.5
  9    986    584 +16.2        832   299      465    10.6    -9.3    +5.1
 10    986    584 +16.2        876   295      492    11.9   -10.8    +5.0
 11    986    584 +16.2        970   320      521    12.7    -8.2    +9.7
 12    986    584 +16.2        966   335      551    13.1    -5.4   +11.9
HOURLY MEAN  2
 13    986    584 +16.2        935   329      585    14.7    -7.5   +12.7
 14    986    584 +16.2        970 D 330      618    14.3    -7.2   +12.4
 15    986    584 +16.2        997   341      657    16.8    -5.5   +15.9
 16    986    584 +16.2        997   346      699    18.0    -4.4   +17.5
 17    986    585 +16.3       1030 D 352      745    19.7    -2.9   +19.4
 18    986    584 +16.2         42     6      794    20.9    +2.1   +20.8
HOURLY MEAN  3
 19    986    585 +16.3       1009     2      846    22.1     +.8   +22.1
 20    986    585 +16.3         49     3      902    23.8    +1.4   +23.7
 21    986    585 +16.3          7     9      965    26.6    +4.3   +26.3
 22    986    585 +16.3         33     7     1032 V  28.3    +3.3   +28.1
 23    986    585 +16.3         36    12       69    29.1    +5.8   +28.5
 24    986    585 +16.3         32    11      135    27.9    +5.5   +27.3
HOURLY MEAN  4
 25    986    585 +16.3         72    18      200    27.4    +8.3   +26.2
 26    986    585 +16.3         43    19      266    27.9    +9.3   +26.3
 27    986    585 +16.3         97    24      335    29.1   +11.7   +26.6
 28    986    585 +16.3         72    29      401    27.9   +13.3   +24.5
 29    986    585 +16.3         67    23      468    28.3   +11.3   +25.9
 30    986    585 +16.3         88    26      539    29.9   +13.2   +26.8
HOURLY MEAN  5
 31    986    585 +16.3         88    30      606    28.3   +14.0   +24.5
 32    986    585 +16.3         57    24      675    29.1   +12.1   +26.5
 33    986    585 +16.3        100    27      744    29.1   +13.0   +26.0
 34    986    585 +16.3         88    32      813    29.1   +15.3   +24.7
 35    986    585 +16.3        128    37      876    26.6   +15.9   +21.4
 36    986    584 +16.2        176    52      943    28.3   +22.4   +17.3
HOURLY MEAN  6
 37    986    584 +16.2        150    56     1005    26.2   +21.8   +14.5
 38    986    584 +16.2        179    57       37    27.0   +22.6   +14.8
 39    986    584 +16.2        161    59      101    27.0   +23.1   +14.0
 40    986    584 +16.2        162    56      171    29.5   +24.4   +16.6
 41    986    584 +16.2        259    74      237    27.9   +26.7    +7.9
 42    986    584 +16.2        231    86      304    28.3   +28.2    +2.0
HOURLY MEAN  7
 43    986    583 +16.2        231    81      374    29.5   +29.1    +4.6
 44    986    583 +16.2        294    92      442    28.7   +28.7    -1.2
 45    986    583 +16.2        254    97      512    32.4   +32.2    -3.7
 46    986    583 +16.2        310    99      591    30.3   +29.9    -5.0
 47    986    583 +16.2        338   115      659    28.7   +26.1   -12.0
```

3725 (beside 40)

3730 (beside 45)

3732 (beside 47)



First clock failure

Fig. 4-9a.

```
        40174 -      46                                                                                         PAGE    103

     19730930
        NR    REF   TEMPERATURE       DIRECTION      VELOCITY       EAST   NORTH          +0        +90       +180      +270      +360               DEGR
3733          CH1     CH2           CH3/5 DEGR     CH4/6 CM/S       COMP    COMP          .0        .20        .40       .60        .80       .100   CM/S
         1    986     581 +16.1      369    125     976    130.8 +106.7   -75.6                      +                                                  V
     HOURLY MEAN  8

        40174 -      46                                                                                         PAGE    104

     19730930
        NR    REF   TEMPERATURE       DIRECTION      VELOCITY       EAST   NORTH          +0        +90       +180      +270      +360               DEGR
              CH1     CH2           CH3/5 DEGR     CH4/6 CM/S       COMP    COMP          .0        .20        .40       .60        .80       .100   CM/S

         2    986     581 +16.1      389    134      25     34.0  +24.3   -23.8                      +   .
3735     3    986     581 +16.1      415    142     111     36.1  +22.1   -28.5                       +   .
3736     4    986     580 +16.1      374    140     191     33.6  +21.7   -25.6                       +  .
```

Second clock failure

Fig. 4-9b.

```
        40174 -     46

   19730930
     NO   REF    TEMPERATURE       DIRECTION       VELOCITY        EAST   NORTH
          CH1    CH2               CH3/5  DEGR     CH4/6  CM/S     COMP    COMP
3737  1   986    580 +16.1         871     217     132   402.6  -241.8  -321.9
      2   986    580 +16.1         289      22     322    78.7   +28.9   +73.2 A
3739  3   986    578 +16.0         222 D    90     620   120.1  +120.1     +.4
   HOURLY MEAN  9
3740  4   986    578 +16.0         987 D    30     705    35.6   +17.8   +30.9
 -    5   986    578 +16.0         743     299      88   170.9  -149.4   +83.1
3742  6   987    578 +16.0         428     204     944   351.8  -140.5  -322.5
      7   986    578 +16.0         449     155    1011    28.3   +12.1   -25.6
      8   986    578 +16.0         490     165      41    26.2    +6.7   -25.3
3745  9   986    577 +16.0         431     162      98    24.2    +7.4   -23.0
   HOURLY MEAN 10
     10   986    577 +16.0         463     158     156    24.6    +9.4   -22.7
     11   986    577 +16.0         450     161     217    25.8    +8.5   -24.4
     12   986    577 +16.0         472     162     272    23.3    +7.1   -22.2
     13   986    577 +16.0         494     170     328    23.8    +4.2   -23.4
3750 14   986    577 +16.0         452     166     381    22.5    +5.3   -21.9
     15   986    577 +16.0         500     167     424    18.4    +4.0   -18.0
   HOURLY MEAN 11
     16   986    577 +16.0         513     178     463    16.8     +.7   -16.8
     17   986    577 +16.0         599     194     502    16.8    -4.0   -16.3
     18   986    577 +16.0         613     210     540    16.4    -8.2   -14.2
     19   986    577 +16.0         693     226     573    14.3   -10.2   -10.0
     20   986    577 +16.0         699     240     606    14.3   -12.4    -7.2
     21   986    577 +16.0         774     254     637    13.5   -13.0    -3.8
   HOURLY MEAN 12
     22   986    577 +16.0         695     253     670    14.3   -13.7    -4.1
     23   986    577 +16.0         816     261     703    14.3   -14.1    -2.3
     24   986    577 +16.0         775     274     739    15.6   -15.5    +1.2
     25   986    577 +16.0         796     271     774    15.1   -15.1     +.2
     26   986    577 +16.0         891     291     811    16.0   -14.9    +5.8
     27   986    577 +16.0         877     306     850    16.8   -13.6    +9.8
   HOURLY MEAN 13
     28   986    577 +16.0         888     305     894    18.8   -15.4   +10.8
     29   986    578 +16.0         889     307     942    20.5   -16.3   +12.4
     30   986    578 +16.0         895     309     992    21.3   -16.7   +13.3
     31   986    578 +16.0         969     323       9    20.9   -12.7   +16.6
     32   986    578 +16.0         945     332      63    22.9   -10.9   +20.2
     33   986    578 +16.0        1000     337     118    23.3    -9.1   +21.5
   HOURLY MEAN 14
     34   986    578 +16.0         958     339     178    25.4    -8.9   +23.8
     35   986    578 +16.0         988     337     241    26.6   -10.3   +24.6
     36   986    578 +16.0        1011     347     308    28.3    -6.5   +27.5
     37   986    578 +16.0         998     348     374    27.9    -5.6   +27.3
     38   986    578 +16.0           9 D   355     475    42.2    -4.0   +42.0
     39   986    578 +16.0          36       8     577    42.6    +5.6   +42.3
   HOURLY MEAN 15
     40   986    578 +16.0          16       9     649    30.3    +4.6   +30.0
     41   986    578 +16.0          67      14     723    31.1    +7.5   +30.2
     42   986    579 +16.1           3      12     793    29.5    +6.0   +28.9
     43   986    579 +16.1          46       8     860    28.3    +4.0   +28.0
     44   986    579 +16.1          54      17     923    26.6    +7.7   +25.5
     45   986    579 +16.1          56      19     986    26.6    +8.5   +25.2
   HOURLY MEAN 16
```

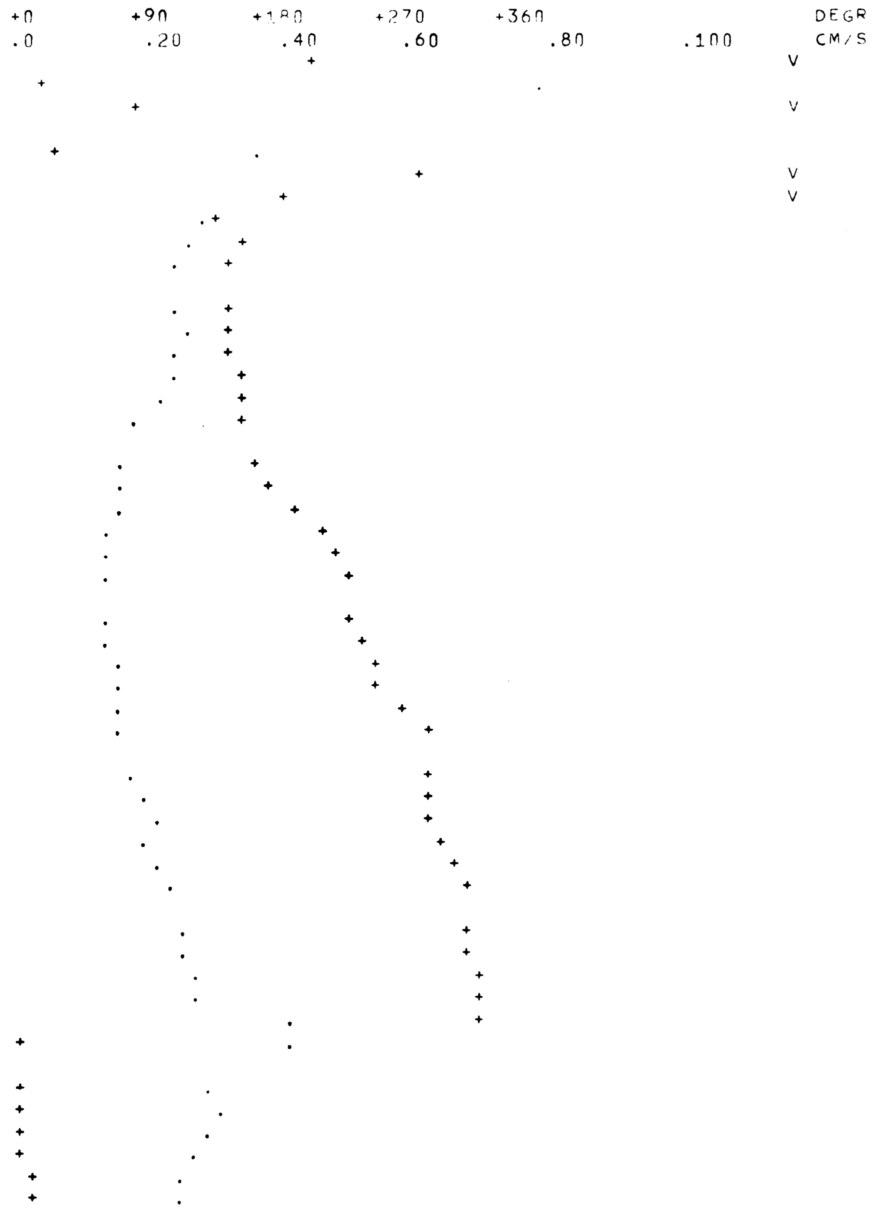


Fig. 4-9c.

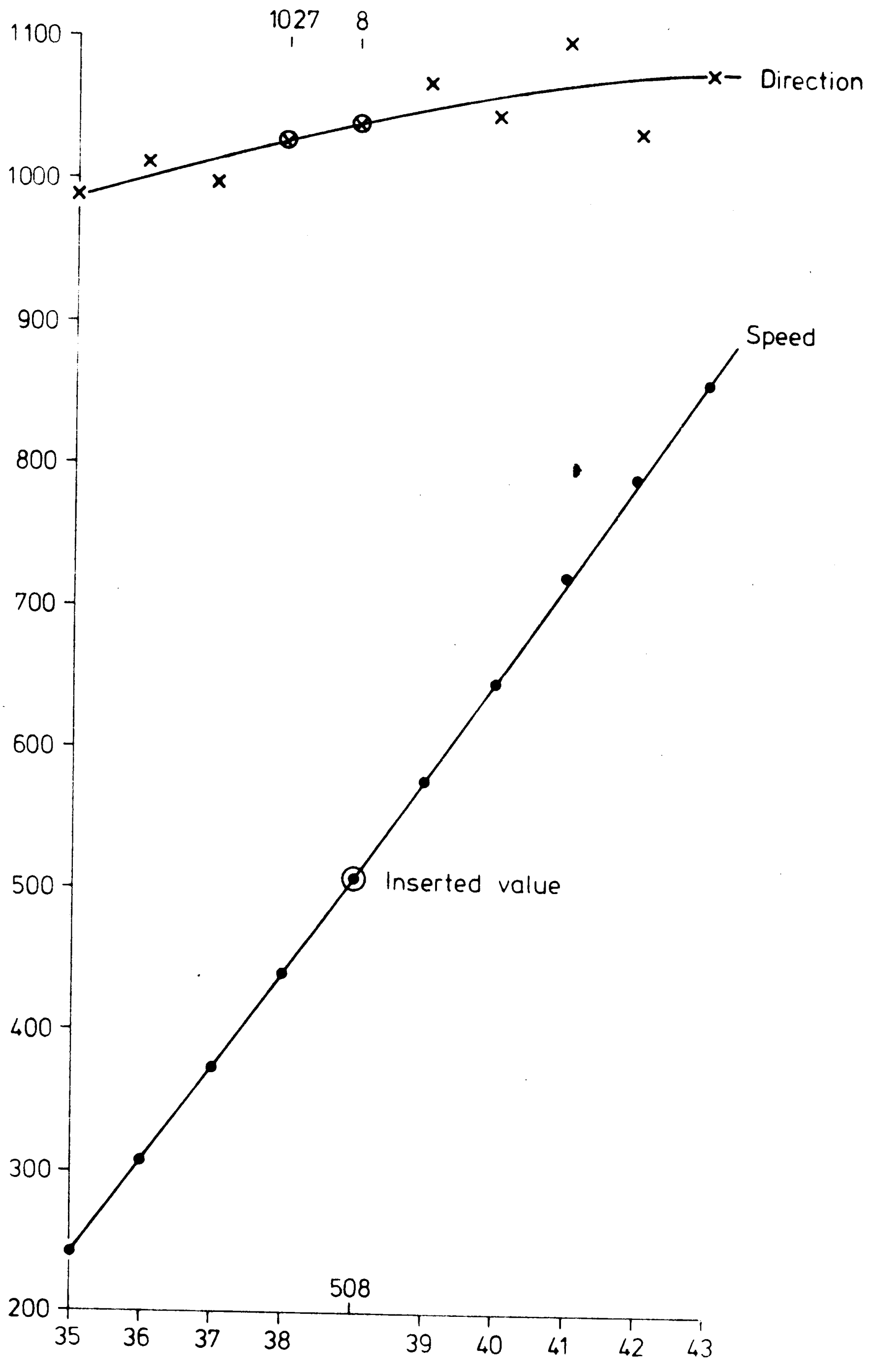


Fig. 4-10. Graphical solution of the irregularity in Fig. 4-9^c at cycle nrs. 38 and 39.

Run time statistics for the other parts of the sample registration are shown in Figs.4-8^b and 4-8^c.

4.3. Visual inspection of the data listing.

The output from the DATA-DECODER and DATA-COMPUTATION programs is subjected to a visual inspection. A complete checklist for this inspection cannot be given, since each current meter has its own peculiarities, which come out in each registration in a different way. Some general statements can however be made about this topic.

First of all the information on the identification, calibration data and times of starting the current meter and recovery must check with the field operations sheet. Mistakes such as in Fig. 4-1 occurring in the recovery time are corrected. This then yields as an expected number of data cycles 4472 or 4473 in stead of 4404 (cf. Fig. 3-8). Comparison of this value with the 4463 decoded cycles starts a number of "speculations" described later in this section.

As a second step detected "clock failures"(if any) are inspected in order to decide whether they are real or can be interpreted in a slightly different way. If the latter is the case some reminder must be written for future use when the testcriteria might be reconsidered.

In the case of our sample registration both detected clock failures are surely real as Figs. 4-9^{a-c} shown.

As a third fase in the visual inspection the "quick look graphs" of speed and direction (cf. Figs. 4-6, 4-9^{a-c}) are scanned for "spikes", needing manual recalculation. As an example let us look at the bottom of Fig. 4-9^c. The interpolation procedure has replaced an original speed reading of 442 by the value 475 (indicated by V). The direction gap value 1023 has been interpolated yielding 9 (indicated by D).
With the test procedure for the speed channel (cf. section 3.4) in mind we learn that in the time series 308, 374, 442, 577, 649, the value 442 is at first accepted, which is fully correct. Then 577 and 649 are rejected as compared to 442, but 649 just falls within the test margin of the series: 374, some rejected value, 577, 649. Thus the reading 442 is considered "suspect" as an afterthought, whereas 577 and 649 are accepted. Plotting the speed data from line nr 35-43 reveals what most probably happened. As shown in Fig. 4-10 the current meter clock somehow missed one data cycle! The graph then gives the value which must be inserted (508). Doing the same thing for the direction channel the value to replace

the gap reading is found (1027) and also the value to be inserted for the missing cycle (8). The east and north components for all data cycles involved in this operation are recalculated and introduced in the paper-tape file of the registration by the program ACM (cf. section 5.2).

Other "spikes" are treated along the same lines and updated according to the situation.

The sample registration shown in Figs. 3-8 and 4-1 through 4-9^{c} is "typical" as far as the program DATA-DECODER is concerned. Also most of the listing made by DATA-COMPUTATION can be considered typical.

As already implicated in the beginning of this section it is not a "typical" registration however. It is not uncommon to find mechanical clocks stopped when recovering current meters. Also stopping and restarting of the clock one or more times during the registration period has been found by our present programs several times. But a difference of only 9 or 10 between the expected number of data cycles and the number actually found is an exception rather than the normal situation.

This particular registration has been made during the JONSDAP 1973 campaign in the Southern Bight of the Northsea. Upon recovery the Aanderaa current meter 986 was found to have stopped due to an exchausted battery. A total of 4463 data cycles has been detected, whereas 4472 or 4473 must be expected. Of the missing 9 or 10 cycles one has already been located and discussed earlier in this section (cf. Fig. 4-10). The fact that the remainder of 8 or 9 cycles is so small and that two clock failures were encountered in close proximity of each other and that the registration looks a bit messy in only a limited range after the second clock failure (Fig. 4-9^{c}), makes it worthwhile to attempt a connection of the three parts of the registration by graphical interpolation.

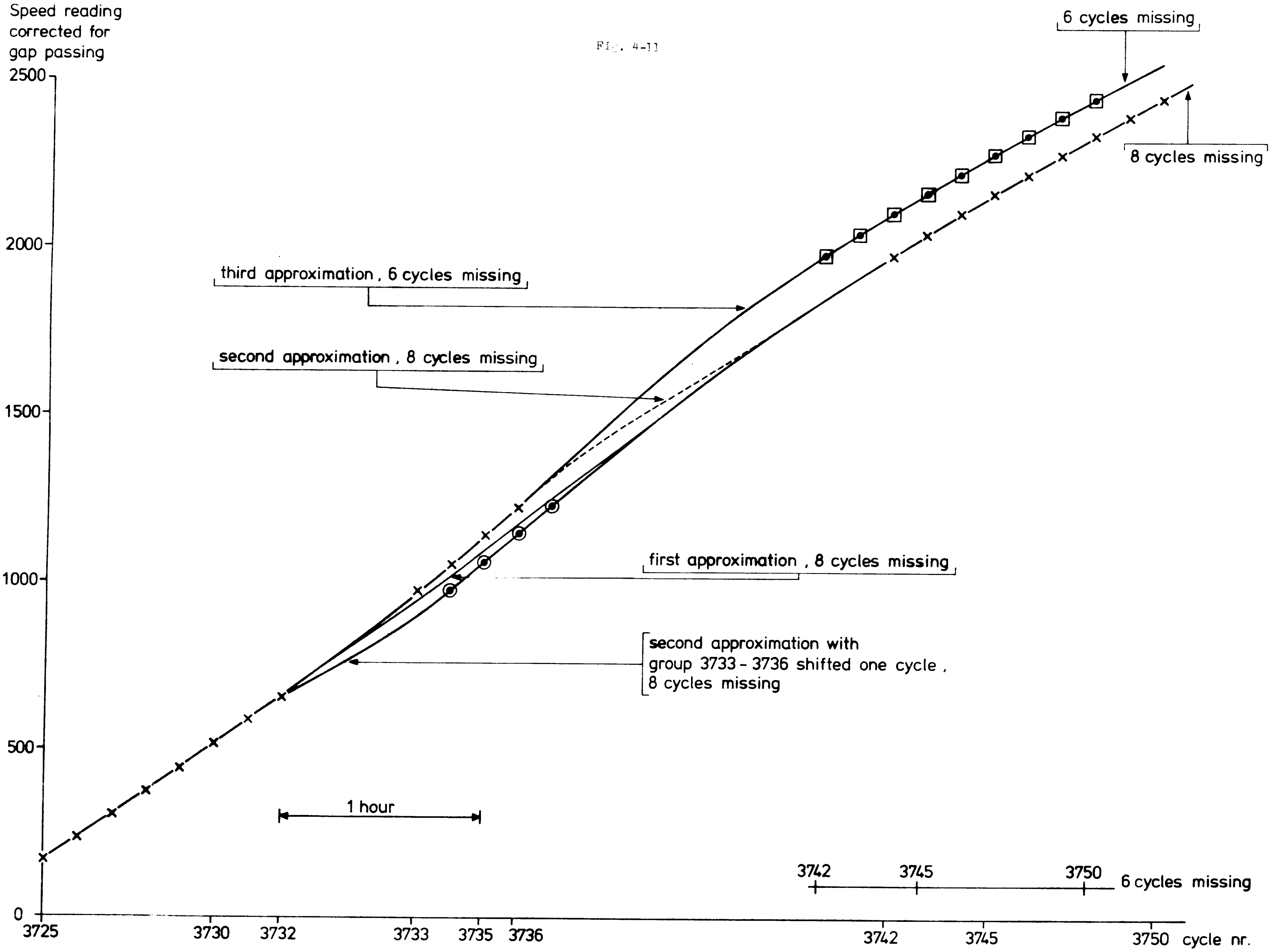
As a first guideline in this procedure we conclude from figs. 4-9^{a-c} that the clock failures are not only real but have also been detected in the proper places.

From fig. 3-8 we furthermore emphasize that the cycle nrs. 3737 and 3741, decoded as translate errors are at least suspect in view of the other cycles in that area.

The interpolation now proceeds as follows (see Fig. 4-11):

- plot the velocity readings of cycle nrs. 3725 - 3732
- plot also the group 3742 - 3750 leaving space for the nine intermediate cycles in the listing plus 8 extra missing cycles

Fig. 4-11


Speed reading
corrected for
gap passing
2500
2000
1500
1000
500
0
3725
3730
3732
3733
3735
3736
3742
3745
3750
cycle nr.
3742
3745
3750
6 cycles missing
1 hour
6 cycles missing
8 cycles missing
third approximation, 6 cycles missing
second approximation, 8 cycles missing
first approximation, 8 cycles missing
second approximation with
group 3733 - 3736 shifted one cycle,
8 cycles missing

- connect these two groups by a smooth line (called first approximation in Fig. 4-11)
- now plot the group 3733 - 3736 in such a way that they best fit to the first approximation, accounting for trends in the various groups (crosses in Fig. 4-11)
- a second approximation curve is now drawn along all data points in the plot.

At this stage we find that the second approximation gives about 3 points where the curvature of the line vanishes corresponding to the same number of extrema in the velocity within a 3 hour period. This is highly improbable in view of the listings preceding and following the area under consideration. Shifting of the 3733 - 3736 group by one time-interval (circles in Fig. 4-11) only enhances this picture.

A solution of this problem is reached by shifting the 3742 - 3750 group by two time intervals (squares in the right top corner of Fig. 4-11) resulting in only one inflection point of this (third) approximation.

In this way we almost certainly have accounted for 6 missing cycles plus one from the spike in Fig. 4-9^c. We therefore conclude that in this registration, covering 31 days, the clock has come to a final stop only 20 - 30 minutes before recovery of the mooring!

The uncertainties, involved however, only justify further use of the registration uptil cycle nr. 3732 the timing of the later cycles being uncertain to ½ - 1½ hours. For areas with relatively strong tidal currents this is too much when one is interested mainly in the small residual currents.

4.4. Summary of the results and timing information.

The main result as implemented in the program DATA-COMPUTATION is the interpolation procedure that givesalmost optimal results complementing the testprocedures in the DATA-DECODER program. This result is mainly due to using a higher order polynomial in the least squares fit computation.

Since for the papertape output a procedure has been written, which suppresses all non-significant characters, the execution time of the program is entirely determined by the speed of the line printer in our installation: a maximum of 10 lines/sec.

The 4460 data cycles of the example shown give a minimum of about 6400 lines of output, not including the identification and calibration data, error listings and runtime statistics.

From Figs. 4-8^{a-c} it can be taken that the error correction took as much as 180 seconds, the data listing 672 seconds including the hourly mean tables.

Total run time was 860 seconds which in a multiprogramming environment with "off-line" printing would probably be less.

5. ACTUAL TIME CURRENT METERS.

This chapter is devoted to the program ACTUAL TIME CURRENT METERS. This program serves the following purposes:

- insertion of manually obtained corrections in the output data of DATA COMPUTATION
- removal of data at the beginning and at the end of the data file when the temperature test did not produce the correct file length (this might e.g. be the case when the rotor has stuck, thus producing nonsense data)
- optional listing of interpolated data at exact ten minute intervals
- computation and listing of hourly mean values centred around the full GMT hours
- computation and listing of tidal mean values for a M2 period ending at 1200 Z, for a M2 period starting at 1200 Z and for a double M2 period centred around 1200 Z
- finally a listing is produced from which a sort of progressive vector diagram can be drawn.

5.1. Initialization and product selection.

The ACM program requires 3 punched paper tapes to be read:

- the identification tape (c.f. Appendix C3, fig. C-6)
- the corresponding output tape from DATA-COMPUTATION
- the correction tape (c.f. Appendix C3, fig. C-7).

Since the campaign and station numbers are punched in the heading of the DATA-COMPUTATION tape, they can be compared with the values of the identification tape. A mismatch forces an immediate termination of the program execution. The only other result of reading these two tapes is the setting of a flag corresponding to the last item in the identification tape. If the value was a 1 a listing of interpolated current velocities at exact ten minute intervals will be produced.

A zero suppresses this listing, whereas any other number (probably due to tape format errors) causes cancellation of the job.

After these small tests a message is printed on the operator's telex to read the correction tape.

5.2. Updating of input data.

The updating of the input data is performed in essentially two steps under control of the correction tape. As outlined in Appendix C3 this tape consists of two parts:

- the first part either contains individual data to be corrected or is empty
- the second part contains information on the number of data cycles to be deleted at the beginning and at the end of the data file.

The first part, when present, is surrounded by the "brackets" -1 and -2, before and after the corrected data.

On detection of -1 as the first item in the correction tape a procedure is entered which expects groups of three numbers:

1st an positive integer number denoting the place of the cycle to be corrected in the data-file

2nd a real number: the corrected east component to be inserted

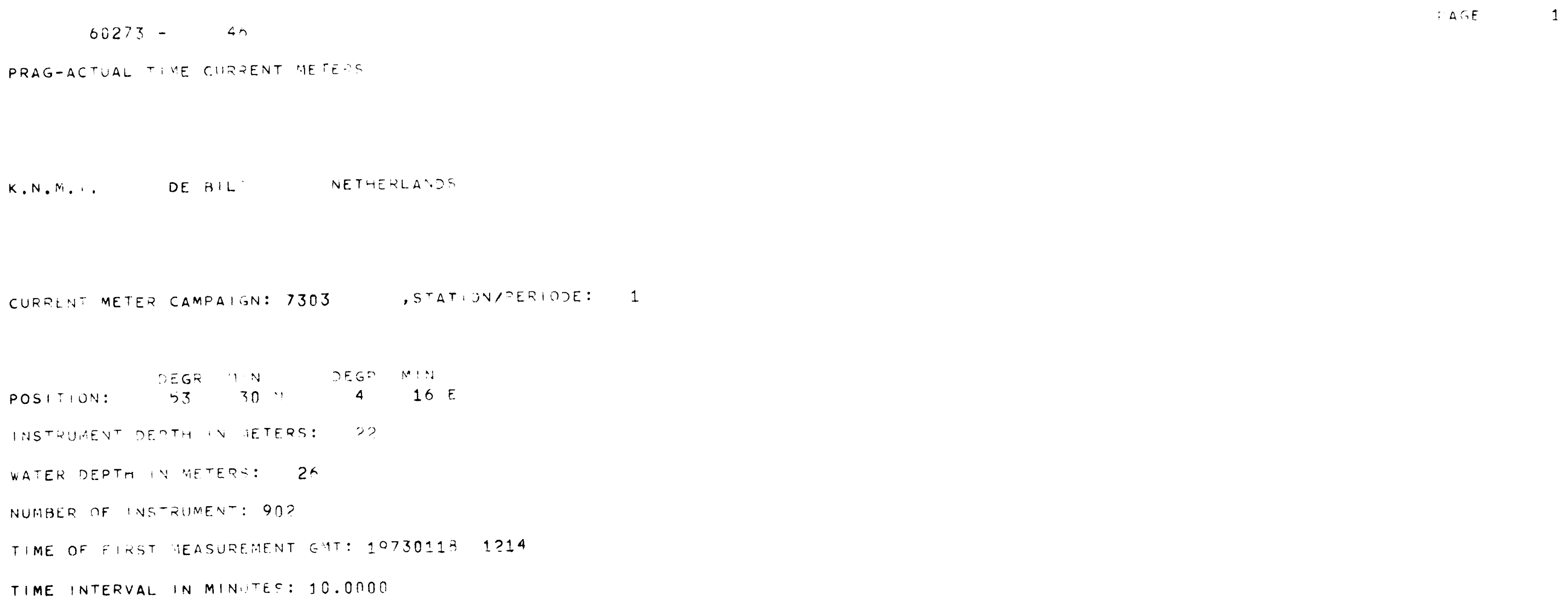
3rd a real number: the corrected north component to be inserted in the indicated place.

These corrections have been derived after visual inspection of the DATA-COMPUTATION listing (c.f. section 4-3). Each corrected cycle is inserted in the indicated place in the data file on the temporary drum storage.

As soon as the first integer number is found to be -2, indicating the end of the corrections, the program execution returns to the main program. Then the last two items in the correction tape are read and processed. Deletion of data at the beginning of the data file is achieved be resetting the origin pointer, at the same time updating the GMT time associated with the data cycle to be considered as the first cycle later on in the processing.

Deletion of data at the end of the file is implemented as a recomputation of the file length.

After these corrections a new file is created which consists of three rather than two items per data cycle, in that the time associated with each data cycle is added to facilitate the data handling during the rest of the program. Moreover the velocity components are adjusted to correct for the actual timestep which might be slightly different from the timestep used in DATA-COMPUTATION due to small discrepancies in the current meter clock. From this so-obtained data-file a papertape is produced, containing the north and east components of the accepted data cycle.



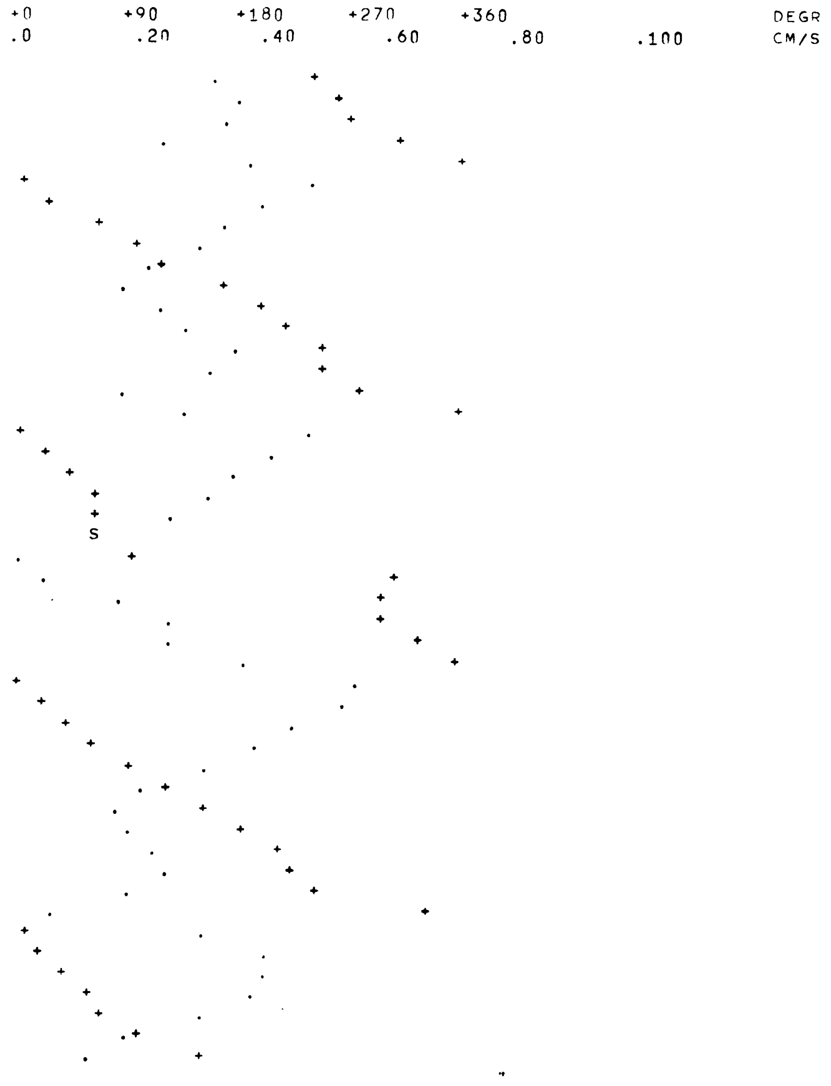
```
       60273 -      46                                                                                  PAGE        2
THE FIRST   11 DATA-POINTS HAVE BEEN SKIPPED

THE REVISED TIME OF FIRST MEASUREMENT IS GMT: 19730118  1404

THE LAST     2 DATA-POINTS HAVE BEEN SKIPPED
```

Fig. 5-2.

```
      60273 -      46

TEN MINUTE VALUES
   DATE     TIME  NORTH   EAST    DIR   VELOC
 YYYYMMDD    GMT   COMP   COMP   DEGR  CM/SEC

19730118    1410  +16.3   -2.8   350    16.5
            1420  +19.8    -.5   359    19.8
            1430  +24.6   +2.2     5    24.7
            1440  +28.9   +3.5     7    29.1
            1450  +31.1   +2.9     5    31.2
            1500  +34.0   +2.8     5    34.1
            1510  +36.7   +2.9     4    36.8
            1520  +38.2   +4.5     7    38.5
            1530  +38.7   +6.3     9    39.2
            1540  +38.8   +7.4    11    39.5
            1550  +37.0  +11.1    17    38.6
            1600  +35.2  +13.4    21    37.7
            1610  +34.2  +14.3    23    37.1
            1620  +32.9  +18.0    29    37.5
            1630  +28.2  +21.0    37    35.2
            1640  +23.9  +22.7    44    33.0
            1650  +22.8  +23.1    45    32.5
            1700  +21.7  +24.7    49    32.9
            1710  +17.2  +28.4    59    33.2
            1720  +11.8  +30.8    69    33.0
            1730   +9.1  +29.7    73    31.1
            1740   +9.2  +28.1    72    29.6
            1750   +8.9  +28.7    73    30.1
            1800   +6.7  +29.0    77    29.8
            1810   +5.3  +27.1    79    27.6
            1820   +2.5  +29.0    85    29.1
            1830    +.5  +28.0    89    28.0
            1840    +.5  +26.2    89    26.2
            1850   -2.5  +23.7    96    23.8
            1900   -6.7  +22.4   107    23.3
            1910   -9.4  +21.8   113    23.7
            1920  -12.2  +20.9   120    24.2
            1930  -11.9  +17.5   124    21.2
            1940  -12.6  +15.9   128    20.3
            1950  -15.3  +14.0   137    20.7
            2000  -17.5  +10.8   148    20.5
            2010  -18.9   +8.2   156    20.6
            2020  -19.1   +5.5   164    19.9
            2030  -20.7   +1.5   176    20.7
            2040  -21.1   -1.0   183    21.1
            2050  -22.3   -1.6   184    22.4
            2100  -23.7   -4.7   191    24.1
            2110  -22.6   -5.6   194    23.3
            2120  -21.7   -6.4   196    22.7
            2130  -20.9  -10.1   206    23.2
            2140  -21.7  -11.6   208    24.6
            2150  -23.4  -11.4   206    26.0
            2200  -22.8  -12.8   209    26.1
            2210  -20.7  -16.5   218    26.5
            2220  -17.7  -18.0   225    25.3
            2230  -18.6  -18.4   225    26.2
            2240  -20.8  -18.6   222    27.9
            2250  -20.4  -21.3   226    29.5
            2300  -18.6  -21.9   230    28.8
            2310  -20.3  -22.6   228    30.4
            2320  -19.4  -26.4   234    32.8
```

Fig. 5-3.

```
      60273 -      46

HOURLY MEANS
  DATE      TIME    NORTH   EAST    DIR    VELOC
YYYYMMDD     GMT     COMP   COMP   DEGR   CM/SEC

19730120       0    -13.5  -28.0    244    31.1
             100     -8.4  -35.6    257    36.6
             200     +1.9  -33.1    273    33.2
             300    +16.4  -16.8    314    23.5
             400    +38.8    -.6    359    38.8
             500    +46.8   +8.0     10    47.5
             600    +34.2  +21.3     32    40.3
             700    +13.1  +30.9     67    33.6
             800     -5.6  +30.2    101    30.7
             900    -12.0  +18.1    124    21.8
            1000    -18.0   +2.5    172    18.2
            1100    -22.0   -8.0    200    23.4
            1200    -20.3  -19.1    223    27.9
            1300    -14.5  -32.0    246    35.2
            1400    -12.9  -29.7    246    32.4
            1500     +2.6  -17.2    279    17.4
            1600    +28.9    -.7    359    28.9
            1700    +46.9  +10.9     13    48.2
            1800    +35.9  +22.4     32    42.3
            1900    +22.9  +28.7     51    36.7
            2000    +12.0  +29.5     68    31.9
            2100     +7.6  +24.3     73    25.4
            2200     +5.4  +11.9     66    13.0
            2300      -.4   +1.8    104     1.9
19730121       0     +4.4   -4.5    314     6.3
             100     +8.7  -16.8    298    18.9
             200    +11.7  -22.6    297    25.5
             300    +24.0  -11.7    334    26.7
             400    +37.0    -.0    360    37.0
             500    +54.7  +12.2     13    56.0
             600    +47.4  +24.9     28    53.6
             700    +33.0  +33.3     45    46.9
             800    +14.3  +36.8     69    39.5
             900     -6.6  +31.3    102    32.0
            1000    -14.6  +16.3    132    21.9
            1100    -16.4   +5.4    162    17.3
            1200    -19.0   -4.2    192    19.4
            1300    -18.8  -13.4    215    23.1
            1400    -15.4  -20.6    233    25.8
            1500     -8.6  -18.9    246    20.7
            1600     +7.8   -2.8    341     8.3
            1700    +30.1  +12.4     22    32.5
            1800    +34.9  +22.4     33    41.4
            1900    +27.3  +32.6     50    42.5
            2000    +16.7  +36.0     65    39.7
            2100     +4.0  +31.5     83    31.7
            2200     -8.5  +19.0    114    20.8
            2300    -12.2   +5.6    155    13.4
```



. speed
+ direction
S speed and direction

Fig. 5-4.

```
       60273 -      46

TIDAL MEANS    24.84 HOURS MEANS
  DATE             MORNING PERIOD                     EVENING PERIOD                  FULL PERIOD
YYYYMMDD  NORTH   EAST    DIR   VELOC   NORTH   EAST    DIR   VELOC   NORTH   EAST    DIR   VELOC
           COMP   COMP   DEGR  CM/SEC    COMP   COMP   DEGR  CM/SEC    COMP   COMP   DEGR  CM/SEC

19730119   +1.1    -.5    336     1.3    +1.9    -.4    348     1.9    +1.5    -.5    343     1.6
19730120   +5.0   -1.4    345     5.2   +10.4   +2.8     15    10.8    +7.7    +.7      5     7.7
19730121  +15.2   +8.3     29    17.3    +2.7   +7.8     71     8.3    +8.9   +8.1     42    12.0
19730122  +11.7   +8.4     36    14.4    +8.2  +14.8     61    16.9    +9.9  +11.6     50    15.3
19730123  +11.3   +4.7     23    12.2    +6.2   +5.9     44     8.5    +8.7   +5.3     31    10.2
19730124   +8.8   +4.5     27     9.9    +2.2   +8.3     75     8.5    +5.5   +6.4     49     8.4
19730125   -1.7   -2.8    239     3.3     +.6   +6.5     85     6.5     -.5   +1.8    106     1.9
19730126   +4.1   +8.2     63     9.2    +2.5   +6.6     69     7.1    +3.3   +7.4     66     8.1
19730127   +6.0  +10.4     60    12.0     +.4   +6.4     87     6.4    +3.2   +8.4     69     9.0
19730128   -4.5   -5.3    230     7.0    -4.0   -5.5    234     6.8    -4.3   -5.4    232     6.9
19730129   +2.7    -.8    344     2.8    +4.1   -2.4    329     4.7    +3.4   -1.6    335     3.7
19730130   +2.4   -2.1    318     3.2    +4.5    +.5      7     4.6    +3.5    -.8    347     3.6
19730131   +2.1   +1.6     38     2.6    +1.3    +.5     22     1.4    +1.7   +1.1     32     2.0
```

Fig. 5-5.

```
      60273 -      46

PROGRESSIVE VECTOR DIAGRAM DATA


  DATE          DISTANCE   TRAVELLED PROGRESSIVE
YYYYMMDD      NORTH      EAST    DIR    DIST     N          E
               COMP      COMP   DEGR    KM

19730119     +1.30       -.39   343     1.36   +1.30       -.39
19730120     +6.64       +.60     5     6.06   +7.94       +.21
19730121     +7.71      +6.99    42    10.40  +15.65      +7.20
19730122     +8.57     +10.04    50    13.20  +24.22     +17.24
19730123     +7.54      +4.58    31     8.83  +31.77     +21.32
19730124     +4.77      +5.52    49     7.29  +36.53     +27.35
19730125      -.46      +1.59   106     1.65  +36.07     +28.93
19730126     +2.86      +6.39    66     7.00  +38.93     +35.32
19730127     +2.74      +7.28    69     7.78  +41.67     +42.60
19730128     -3.68      -4.68   232     5.95  +37.99     +37.92
19730129     +2.93      -1.36   335     3.23  +40.92     +36.56
19730130     +2.99       -.69   347     3.07  +43.91     +35.87
19730131     +1.44       +.92    32     1.71  +45.35     +36.78
```

Fig. 5-6.

```
       60273 -      46                                                                                                    PAGE      50

START      INLEES      WIGWAG      OMZET      PONSEN      TENMIN      UURGEM      TIJGEM      PLOT
      2.68        56.62      223.20      252.10      549.66      759.77      797.29      800.30      801.91


EINDE PRAG-ACTUAL TIME CURRENT METERS
```

Fig. 5-7.

5.3. Explanation of the output.

In fig. 5-1 - 5-7 a sample output is shown, which is virtually self-explanatory.

A few remarks, however, must be made here.

- The (optional) listing of ten minute values is obtained by linear interpolation of the components between the two adjacent data cycles.
- The hourly mean values are computed from an integration of the components from H-30 to H+30 where H is the time (full GMT hours) in the listing. Linear interpolation is used where necessary.
- Tidal means are calculated by integration of the velocity components from 1200 Z - 12.42 hours to 1200 Z for the morning period and from 1200 Z to 1200 Z + 12.42 hours for the evening period.
- "Daily" means are obtained by averaging of the results for the morning and evening periods.
- The progressive vector diagram data are obtained by proper scaling of the "daily" means to reduce the integration over a double M2 (24.84 hours) period to a displacement during 24 hours.
  These displacements are shown separately and cumulative (c.f. fig. 5-6).

5.4. Timing information.

On the Electrologica X8 installation of the K.N.M.I. the job runtime of the ACM program for a typical 6 week registration is about 20 minutes.

The ten minute listing option takes about 12 minutes extra.
In a multiprogramming environment CPU times could be about 3 minutes, since I/O takes most of the time.

6. Possibilities for future extension and modification.

Although a good deal of all available experience obtained during the development of this software package has been incorporated in the programme versions presented, some errors might occur under unusual circumstances.

Statistics for the sort of these errors and their causes cannot be given, due to the fact that most of them occurred only once, without any clearly distinguishable reason.

Since a change of the K.N.M.I.'s computer installation will occur in the near future, due regard must be given to the transscription of the programmes. Most certainly a complete rewrite starting from the present experience is the way to obtain the optimum result.

As far as future extensions are concerned the following suggestions can be made.

- The decoding procedure has been devised for the standard Plessey and Aanderaa sampling schemes. Other sampling schemes could in principle rather easily be incorporated if a "syntax" for the input can be defined (c.f. section 3.2, Table 3-III).
- Other tests could be built. A useful one is a test for errroneous (single) temperature readings. A corresponding interpolation procedure should then be added as well.
- For instrument tests and "quick look" data inspection the inclusion of spectra and/or histograms could be a useful tool.
- An algorithm for finding erroneous readings based on dynamical test criteria would be an improvement. As an example the criterion for the direction channel can clearly be more stringent during periods of calm weather than during storms. By taking the distribution of direction changes during a day (or less) the criterion can be adapted to the situation to be tested.
- A tidal analysis, at least for the main components, would be a valuable extension. This is especially the case if one is studying e.g. the variability of tidal "constants" due to meteorological forcing.
- The way of data storage is at present in fact dictated by the limitations of our computer installation. If faster intermediate storage (e.g. on magnetic tape or disk pack) is available more elaborate and more detailed data processing could be considered than is feasible with punched paper tape as a storage. This applies e.g. to experiments on

the test criteria used. An extra programme step could as an example produce a suggestion for the criteria to be used on the particular data series. After (off-line) visual inspection the actual tests and correstions can be performed etc.

Of course the above points are merely suggestions, which could be implemented as additions to the modular programmes. They are only given as an illustration of the potential of the software discussed in this report.

The proper choice of the modules suggested (or others) will highly depend on the use that is to be made of the current meter data obtained and the area where they come from.

References.

1. de Crook, Th. and van der Veen, K., "Beschrijving van de verwerking van meetresultaten van zelfregistrerende stroommeters", KNMI internal report V-241 (1972).

2. Foster, J.M., "Automatic syntactic analysis" (MacDonald, London, 1970).

3. Koster, C.H.A., "A compiler compiler", Mathematisch Centrum, Report MR 127/71, (Amsterdam, 1971).

Appendix A.

Summary of the EL-X8 I/O functions used in the programs DATA-DECODER-MULTIPURPOSE, DATA-COMPUTATION and ACM.

Paper tape reader. Can handle 5, 7 or 8 track tape.

READ — gives the value of the next number (real or integer) on tape;7-track flexowriter code is assumed.

REHEP — gives the binary value of the next character on the tape. The decoding must be done in the user program.

Line printer.

NEWPAGE — skip to the first position of the first line on the next page.

NLCR — skip to the first position of the next line ("new line and carriage return").

CARRIAGE(n) — equivalent to n successive NLCR instructions.

SPACE(n) — skip n positions on the current line (i.e. "print" n blanks).

PRINTTEXT (≮ text ≯) — print the characterstring "text"

FIXT (n, m, x) — n and m are integers
x may be either integer or real;print the number x with a maximum of n digits before and m digits after the decimal point. Leading zero's are replaced by blanks; the sign immediately preceeds the most significant digits or the decimal point. A blank is inserted after the number.

ABSFIXT (n,m,x) — as FIXT (n, m, x) but with a blank instead of the sign.

Operator's telex ("COTEL").

TELETEXT (≮ text ≯)   print the characterstring "text" on the COTEL as a seperate line.

Paper tape punch.

Gives 7-track paper tape.

RUNOUT   gives a length of blank tape

PUHEP(n)   punch a character with the binary value n

The next paper tape punch functions give 7-track flexowriter code.

PUNLCR   punch a NLCR chracter ("new line and carriage return")

PUTEXT (≮ text ≯)   as PRINTTEXT (≮ text ≯) but on the paper punch

FIXP (n, m, x)   as FIXT (n, m, x) but on the paper punch

ABSFIXP (n,m,x)   as ABSFIXT (n,m,x) but on the paper punch

Temporary (drum) storage.

inarray (drum, adres, A)

adres is an integer
A an array that may be either real or integer and may have any number of dimensions and may be of any length.
The type, dimension and length information are taken from the declaration of the array A.

The array A is read in from drum starting at the drum-address adres.
It should be emphasized that an integer occupies 1 memory word and a real 2 memory words (of 27 bits each) and that the drum-storage can be looked at as a core-image.

outarray (drum, adres, A)

As inarray (drum, adres, A) but now the array A is copied on the drum-storage starting at address adres.

hold (A)

A is an array
The functions inarray and outarray initialize the transfer of the appropriate number of words in the direction indicated and then return control to the main program. The actual transfer is done on a databreak basis and independent of the main program. The function hold (A) acts as a waitloop until all transfers to or from A are finished.

REMAINDER (m,h)

A machine code function which computes the remainder of the integer division:
m/n

Appendix B.

Description of the initial and calibration data area on drum storage.

In Table 2-I the general layout of the temporary storage has been given. In the tables BI-BIV of this Appendix a detailed description of the contents of the arrays ADMIN, CYCLESTORE, CALIB and COMPASS is presented.

TABLE B-I

Contents of integer array ADMIN

drum address	element number	contents name	description
0	1	campnr	campaign number; part of identification of the registration
1	2	statnr	station or period number;
2	3	lat	latitude; + north - south
3	4	long	longitude; + east - west
4	5	depth	depth of instrument below seasurface at the time the mooring is laid
5	6	wdepth	waterdepth at the time the mooring is laid
6	7	type	instrument type: 11 Plessey, reference in track 7, 12 Plessey, reference in track 8 21 Aanderaa, " " " ", 22 Aanderaa, " " " "
7	8	metno	instrument identification number
8	9	function	option code. Explanation is given in section 3.1
9	10	spare	
10	11	timestep	time interval of measurement in minutes
11	12	edag	date of first data-point on deck ("date of laying the mooring"); GMT
12	13	ldag	date of last data-point on deck ("recovery data"); GMT
13	14	spare	
14	15	spare	
15	16	gap	equivalent of the gap in the speed potentiometer
16	17	spare	
17	18	spare	
18	19	spare	
19	20	spare	
20	21	s	hourly period the instrument is in place; result of temperature test or first period (section 3.3); GMT
21	22	dag	date period the instrument is in place; result of temperature test or edag (section 3.3); GMT
22	23	spare	
23	24	spare	
24	25	spare	

TABLE B-II

Contents of integer array CYCLESTORE

drum address	element number	contents name	description
25	1		these locations are used eventually to store the number of the data-points where a clock stop has been detected in the test of the speed channel (cf. section 3.4)
26	2		
27	3		
28	4		
29	5		
30	6		
31	7		
32	8		
33	9		
34	10		
35	11	endcycle	number of data-points until either the end of the whole registration or recovery (cf. section 3.3) is detected or too many clock stops are suspected (cf. section 3.4), whichever comes first
36	12	firstcycle	number of the data-point at which the instrument is detected to be in place according to the temperature test or 1 if either no temperature test is performed or the temperature test was unsuccesfull (cf. section 3.3)
37	13	totalcycle	total number of data-points in the registration
38	14	storecount	number of blocks in the DATA-file
39	15	nrexpected	number of data-points expected as computed from begin and end times (bm and em)

TABLE B-III

Contents of real array CALIB

drum address of first word	element number	contents name	description	
40	1	ao	constants in the fit to temperature calibration data:	
42	2	a1	temperature (^{o}C) = ao + a1 * digital resistance value	
44	3	kf	multiplying constant in direction calculation; for Plessey and Aanderaa 0.345	
46	4	kv	adding constant in direction calculation (compass variation)	
48	5	apro	minimum speed at which propellor/rotor starts rotating	
50	6	cfs	conversion factor for the speed computation from digital counts/min to cm/sec	
52	7	spare		
54	8	spare		
56	9	spare		
58	10	spare		
60	11	bo	constants of the salinity calibration:	for Aanderaa meter only
62	12	b1	salinity (o/oo) = bo + b1 * digital resistance value	
64	13	co	constants of the depth calibration:	
66	14	c1	depth (m) = 10 * (co + c1 * digital resistance value)	
68	15	spare		
70	16	spare		
72	17	spare		
74	18	spare		
76	19	spare		
78	20	spare		
80	21	etijd	time of first data-point on deck before laying of the mooring; GMT	
82	22	ltijd	time of last data-point on deck after recovery of the mooring; GMT	
84	23	tijd	time the instrument is in place according to tamperature test or etijd (cf. section 3.3); GMT	
86	24	spare		
88	25	spare		

TABLE B-IV

Contents of real array COMPASS			
drum address of first word	element number	contents name	description
90	0		compass correction to be applied at 0° reading
92	1		" " " " " " 45 "
94	2		" " " " " " 90 "
96	3		" " " " " " 135 "
98	4		" " " " " " 180 "
100	5		" " " " " " 225 "
102	6		" " " " " " 270 "
104	7		" " " " " " 315 "
106	8		" " " " " " 360 "

Appendix C.

Input documents and convemtions.

C1. Forms for the DATA-DECODER-MULTIPURPOSE program.

Two forms are used for the preparation of the first input paper tape for the program DATA-DECODER.

In fig. C-1 the form for Plessey-type current meters is shown; in fig. C-2 the Aanderaa-form.

Apart from the two constants bm and em the data are read into the appropriate array-elements as shown in Tables BI - BIV in Appendix B.

The constants bm and em are the combined data and time of the first data-point before laying and the last data-point after recovery respectively as a 12 digit number. For instance 10.15 GMT on April 25, 1973 is signalled to the program thus: 197304251015. bm is then decoded to give edag and etijd (19730425 and 1015 respectively in the above example). em in the same way gives ldag and ltijd.

The "default" values for s, dag and tijd (cf. Tables BI and BIII) are set to resp. 10, 19730425 and 1015.

Note that the tests as described in sections 3.3 and 3.4 of course may result in other values for s, dag and tijd.

From the forms a 7-track flexowriter code free format paper tape is prepared.

Here again it should be noted that a subsequent program such as DATA-COMPUTATION, is selected according to the value of werk. This means that no additional input tapes are needed for that second program.

C2. Preparation of the 8-track translator tape.

The magnetic tape as it comes from the current meters is translated using the Plessey translator giving a 8-track paper tape (cf. fig. C-3).

On the magnetic tape each registration cycle is terminated by a so-called reference pulse which is punched in the first character of the next sequence. As a consequence this reference bit must be added manually to the first character of the first data-cycle on the punched papertape, as illustrated in figs. C-4a and C-4b.

(Note that the tracks 7 and 8, parity and reference, have been interchanged in our apparatus to have the reference bit in channel 8 to facilitate visual inspection of the tape.)

At the end of the punched papertape one or more erase characters are inserted (fig. C-5) as an "end of registration" mark which is sensed by the input buffer routine.

C3. Forms for the ACM program.

For ACM also two forms exists to prepare inputtapes, which are all 7-track flexowriter code and in free format.

Fig. C-6 shows the form for the input of constants. The numbers 1 trough 7 have the same meaning as in the DATA-DECODER case, the other data are new or have a slightly different meaning.

To get the proper value for edag and etijd one should proceed as follows. The first data-cycle on the output tape of DATA-COMPUTATION corresponds to the mean speed value between the first data-point with the instrument in its place and the next data-point. The direction is computed by averaging the directions of this first data-point and its successor. Therefore we decided to take as the time (and date) of the first data-cycle in the outputtape of DATA-COMPUTATION the time (and date) as found in DATA-DECODER for tijd (CALIB [23], cf. Table B-III) and dag (= ADMIN [22] , cf. Table B-I) increased by half the time interval v. For instance, if tijd = 1025 GMT and v = 10, etijd in ACM will be 1030 GMT.

The value for deltat is found by dividing the time between the first and the last data-point on the magnetic tape by the number of data-points, tw, minus 1.

The boolean tien is set true if a 1 at the end of the input tape signals, that a calculation and printing of interpolated velocity values at H + 00, H + 10, H + 20, H + 30 etc is needed. A zero, meaning no ten minute values wanted, will set tien to false. A number other than 0 or 1 will terminate the program execution.

A second form (see fig. C-7; cf. section 5.2) is used to prepare the correction tape. This is divided in two parts: one to have the possibility to insert manually computed data-cycles in the registration as given in the output tape of DATA-COMPUTATION and a second to skip a number of data-cycles at the beginning and/or the end of the registration.

Formulier voor de voorloopband van het Plessey computerprogramma.

Dit formulier in te leveren met de ponsband, zoals deze uit de vertaalmachine komt, bij H.W. Riepma, kamer 240, tel. 268.

1. Campaign
2. Station
3. Latitude
4. Longitude
5. Instrument-depth in meters
6. Waterdepth in meters
7. Type of instrument
8. Number of instrument
9. Time first measurement
10. Time last measurement
11. Time interval in minutes
12. Temperature: zero
13. Temperature: constant factor
14. Compass: variation
15. Compass: constant factor
16. Corrections (compass) for directions: 0 (45) 360

17. Propellor: minimum speed
18. Velocity: constant factor
19. Gap
20. Function

1.		campnr
2.		statnr
3.		lat
4.		long
5.		depth
6.		wdepth
7.		type
8.		metno
9.		bm
10.		em
11.		v
12.		a0
13.		a1
14.		kv
15.		kf
16.		
17.		apro
18.		cfs
19.		gap
20.		werk

Fig. C-1. Form for DATA-DECODER input tape for Plessey current meter.

Formulier voor de voorloopband van het Aanderaa-computerprogramma.

Dit formulier in te leveren met de ponsband, zoals deze uit de vertaalmachine komt, bij H.W. Riepma, kamer 240, tel. 268.

1.	Campaign	1.		campnr
2.	Station	2.		statnr
3.	Latitude	3.		lat
4.	Longitude	4.		long
5.	Instrument-depth in meters	5.		depth
6.	Waterdepth in meters	6.		wdepth
7.	Type of instrument	7.		type
8.	Number of instrument	8.		metno
9.	Time first measurement	9.		bm
10.	Time last measurement	10.		em
11.	Time interval in minutes	11.		v
12.	Temperature: zero	12.		a0
13.	Temperature: constant factor	13.		a1
14.	Salinity: zero	14.		b0
15.	Salinity: constant factor	15.		b1
16.	Pressure: zero	16.		c0
17.	Pressure: constant factor	17.		c1
18.	Compass: variation	18.		kv
19.	Compass: constant factor	19.		kf
20.	Corrections (compass) for directions: 0 (45) 360	20.		
21.	Propellor: minimum speed	21.		apro
22.	Velocity: constant factor	22.		cfs
23.	Gap	23.		gap
24.	Function	24.		werk

Fig. C-2. Form for DATA-DECODER input tape for Aanderaa current meter.

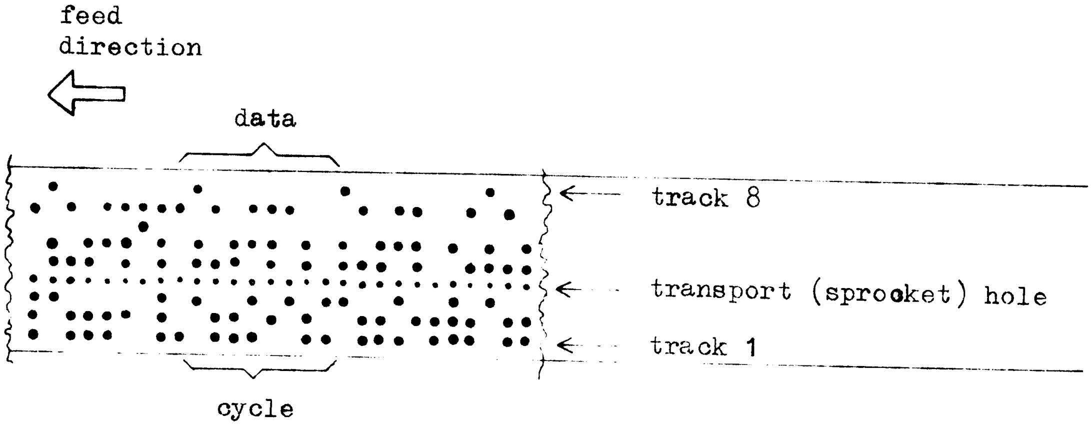


Fig. C-3 Part of 8-track tape for Plessey MO21 current meter

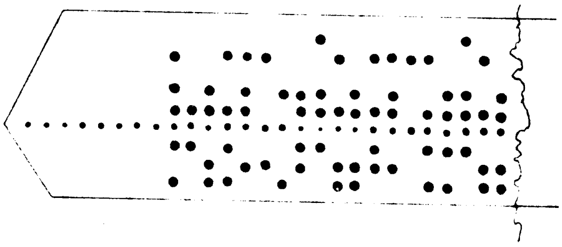


Fig. C-4a Start of 8-track tape as produced by translator.

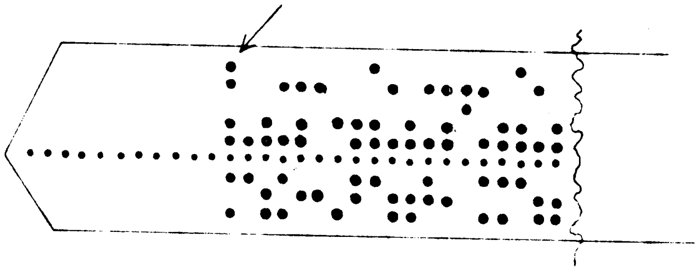


Fig. C-4b Same as Fig. C-4a but with first reference pulse manually inserted.
(Note: in this specific case a parity error will be signalled in the first character).

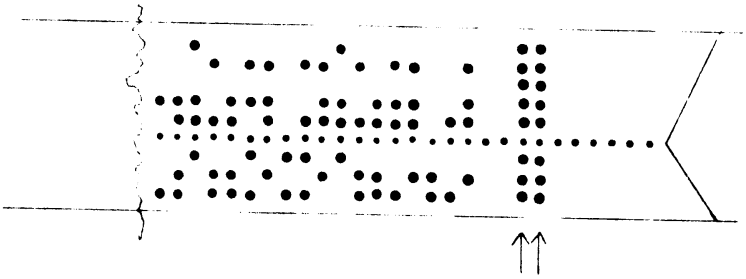


Fig. C-5 End of tape mark with at least one erase character. (Note: blank tape is skipped).

Formulier voor de voorloopband van het ACM-programma.

'campagne-nummer, voorloopband'

	'	, voorloopband'	
1. Campaign	1.		campnr
2. Station/period	2.		statnr
3. Latitude	3.		lat
4. Longitude	4.		long
5. Instrument-depth in meters	5.		depth
6. Waterdepth in meters	6.		wdepth
7. Number of instrument	7.		metno
8. Date	8.		edag
9. Middle time of first measuring period (GMT)	9.		etijd
10. Time interval in minutes	10.		deltat
11. Ten minute values: yes = 1 no = 0	11.		tien

Fig. C-6. Form for the ACM input tape.

Formulier voor de correctieband (G2) van het ACM-programma.

1. Begin mark corrections

2.a. number of measurement to be corrected
b. corrected east component

c. corrected north component

a.

b.

c.

3. End mark corrections

4. Number of measurement to be skipped at the begin
5. Number of measurement to be skipped at the end

'campagne-nummer, G2'

'	, G2'
1.	-1
2.a.	
b.	
c.	
a.	
b.	
c.	
3.	-2
4.	
5.	

Fig. C-7. Form for the ACM correctionstape.

The correction part proper is marked by -1 and -2 at the begin and the end of the section respectively. Between these marks the corrections are grouped in the order: number of the measurement to be corrected, new east component (cm/sec), new north component (cm/sec). The number of the measurement is obtained by "counting" in the DATA-COMPUTATION output; the new components should be calculated after visual inspection of that output from whatever pertinent information is available in the case.

If no corrections need be applied the -1, -2 marks may be dropped or punched without any effect in either case, except that a subroutine call is generated when punched, which subroutine immediately returns control when reading the -2.

The skip section should always be present in the form of two non-negative integers.

The exact effects are described in section 5.2. The values are obtained by the manual and visual inspection of the DATA-COMPUTATION output, combined with the mooring logbook information.

Appendix D.

DATA-DECODER-MULTIPURPOSE

Algol-60 source listing.

```

740206:

BEGIN   COMMENT K.N.M.I.-PRAG-DATA-DECODER-MULTIPURPOSE:

        INTEGER VERSIE:
        BOOLEAN ARRAY PAR[0:255];
        VERSIE:=740206:

        COMMENT DECODEERT DE 8-GATS-PONSBAND AFKOMSTIG VAN EEN ZELFREGISTREREFNDE STROOMMETER,
        TEST OP FOUTEN IN STRUCTUUR VAN DE INVOERSTROOM,
        EN BOUWT DE BENODIGDE BESTANDEN VOOR DE VERDERE BEWERKING OP DE TROMMEL OP.
        AFHANKELIJK VAN HET SLUITGETAL IN DE VOORLOOPBAND WORDT EEN KEJZE UIT DE
        TEST MOGELIJKHEDEN EN DE VOLG-PROGRAMMA'S GEMAAKT:
                0       TEST OP TEMPERATUUR, SNELHEID EN RICHTING,
                        GEVOLGD DOOR DATA-COMPUTATION ZONDER PONSBAND-JITVOER
                1       ALS 0, DOCH MET PONSBAND-UITVOER
                2       ALS 1, DOCH ZONDER TEMPERATUUR-TEST
                3       ER WORDT NIET GETEST IS BEDOELD VOOR, VERWERKING VAN
                        INSTRUMENT TESTBANDEN
                4 T/M 8 ALLEEN DECODEREN,VOLG-PROGRAMMA'S NADER TE BEPALEN
                9       ALLEEN DECODEREN GEVOLGD DOOR DATA-PRINTTAPE

        DE BENODIGDE ADMINISTRATIE WORDT EVENEENS VIA DE TROMMEL DOORGEGEVEN NAAR HET VOLG-PROGRAMMA
        DAT DE GEWENSTE BEREKENINGEN UITVOERT, DIRECT  AANSLUITEND AAN
        DATA-DECODER:

        BEGIN   COMMENT IN DIT BLOK WORDT PAR GEVULD VOLGENS DE METHODE VAN BOOE-FFTWAVE-PROGRAMMA'S;
                INTEGER I,J,K,L,BINC;
                INTEGER ARRAY BIN[0:7]:

                FOR I:=1 STEP 1 UNTIL 254 DO PAR[I]:=TRUE;
                FOR I:=0 STEP 1 UNTIL 7 DO BIN[I]:=2↑I:
                COMMENT 0 EN 8 PONSINGEN; PAR[0]:=PAR[255]:=FALSE;
                COMMENT 2 EN 6 PONSINGEN;
                FOR I:=0 STEP 1 UNTIL 6 DO
                        FOR J:=I+1 STEP 1 UNTIL 7 DO
                                BEGIN BINC:=BIN[I]+BIN[J]; PAR[BINC]:=PAR[255-BINC]:=FALSE; END;

                COMMENT 4 PONSINGEN;
                FOR I:=0 STEP 1 UNTIL 4 DO
                        FOR J:=I+1 STEP 1 UNTIL 5 DO
                                FOR K:=J+1 STEP 1 UNTIL 6 DO
                                        FOR L:=K+1 STEP 1 UNTIL 7 DO
                                                PAR[BIN[I]+BIN[J]+BIN[K]+BIN[L]]:=FALSE;
        END VULBLOK PAR;

        BEGIN COMMENT IN DIT BLOK WORDEN ALLE GROOTHEDEN EN PROCEDURES GEDECLAREERD EN HET DECODEREN GEDAAN:
                INTEGER ARRAY TWBR[1:15],WIG,WAG[1:200], TYD[1:5], A[1:5];
                BOOLEAN BLIST, AANDERAA, FASE;
                REAL BM,EM:
                INTEGER V,H,ADRES,I,J,N,SOMPAR,ADR,VGT,VGR,VGS,SDADR,TCTAL,TEND,REGEL,
                        LOOP,TW,S,CHAR,MEANT,TWFIRST,DAG,PERF,JUMP;

                INTEGER TYPE,TZTEL,WERK;
                REAL TIJD,RHULP;
                INTEGER METNO,VERTAAL;

                PROCEDURE BREEKAF(TEXT); STRING TEXT:
```

```
                BEGIN TELETEXT(‹PRAG-DATA-DECODER WEGENS FOUTEN GESTOPT, VOLG-PROGRAMMA HOEFT NIET GESTART›):
                        CARRIAGE(4);
                        PRINTTEXT(TEXT);
                        CARRIAGE(4);
                        GOTO STOP;
                END BREEKAF;

                INTEGER PROCEDURE REMAINDER(M,N); VALUE M,N; INTEGER M,N:
                        ‹128,21495808,128,4194365,54,54,128,5554175,128,-42369786,128,
                        22167550,128,4784128,128,6914048,128,51396607,128,13647870,128,
                        -28262401,128,-64487421,128,-44401920›:

        REAL PROCEDURE RMOD(REAL,GETAL); VALUE REAL,GETAL; REAL REAL,GETAL;
        BEGIN   RMOD:=REAL-ENTIER(REAL/GETAL)*GETAL END;

        PROCEDURE TIMING(TIJD); REAL TIJD;
        BEGIN   IF RMOD(TIJD,100)≥60 THEN TIJD:=TIJD+40;
                IF TIJD≥2400 THEN BEGIN TIJD:=TIJD-2400; DAG:=DAG+1 END;
        END TIMING:

        PROCEDURE DAGTEL:
        BEGIN   INTEGER HDAG,MND;
                HDAG:=REMAINDER(DAG,100);
                IF HDAG≤28 THEN GOTO DAGKLAAR;
                MND:=REMAINDER((DAG-HDAG)÷100,100); IF MND=2 THEN
                BEGIN   IF REMAINDER(DAG÷10000,4)=0 THEN
                        BEGIN   IF HDAG=30 THEN DAG:=DAG+71; GOTO DAGKLAAR END
                        ELSE BEGIN DAG:=DAG+72; GOTO DAGKLAAR END;
                END MND=2;
                IF MND=1 ∨ MND=3 ∨ MND=5 ∨ MND=7 ∨ MND=8 ∨ MND=10 ∨ MND=12 THEN
                BEGIN   IF HDAG=32 THEN
                        BEGIN   DAG:=DAG+69; IF MND=12 THEN DAG:=DAG+8800; END:
                        GOTO DAGKLAAR:
                END;
                IF HDAG=31 THEN DAG:=DAG+70;
        DAGKLAAR:
        END DAGTEL:

        PROCEDURE OPTIONS;
        BEGIN COMMENT   PRINT DE OPTION-LIST;
                SWITCH PRINTER:=0,1,2,3,4,5,6,7,8,9;
                PRINTTEXT(‹OPTIONS USED:›); NLCR; NLCR:
                PRINTTEXT(‹PROGRAM DECODER›); SPACE(25);
                PRINTTEXT(‹SUCCESSOR PROGRAM›); CARRIAGE(4);
                GOTO PRINTER[WERK+1];
        0:      PRINTTEXT(‹TEMPERATURE,›); SPACE(28);
                PRINTTEXT(‹DATA-COMPUTATION WITHOUT OUTPUT-TAPE›);
                GOTO 10;
                2:      SPACE(40); GOTO 12;
        1:      PRINTTEXT(‹TEMPERATURE,›); SPACE(28);
        12:     PRINTTEXT(‹DATA-COMPUTATION WITH OUTPUT-TAPE›);
        10:     NLCR; PRINTTEXT(‹DIRECTION AND SPEED TEST›); NLCR;
                GOTO END OPTIONS;
        9:      PRINTTEXT(‹PRINTTAPE›): NLCR;
                GOTO ENDOPTIONS;
        3:      PRINTTEXT(‹INSTRUMENT TEST›); NLCR; GOTO END OPTIONS;
        4:5:6:7:8:
        END OPTIONS:
        END:

```

```
PROCEDURE ASKNEXT;
BEGIN COMMENT VRAAGT VIA DE CITEL HET DOOR ADMIN[9] GEGEVEN VOLGPROGRAMMA AAN;
            SWITCH CNTRL:=0,1,2,3,4,5,6,7,8,9;
            GOTO CNTRL[WERK+1];
0:1:2:3:         TELETEXT(‹WILT U DE P-BAND DATA-COMPUTATION INLEGGEN›);
            GOTO END ASKNEXT;
4:5:6:7:8:
9:
END ASKNEXT;
END;


PROCEDURE KOP;
BEGIN   PRINTTEXT(‹   NR  REF  TEMPERATURE    DIRECTION    VELOCITY       EAST  NORTH›); SPACE(16);
        PRINTTEXT(‹+0        +90       +180      +270     +360                  DEGR›); NLCR;
        PRINTTEXT(‹        CH1    CH2            CH3/5 DEGR    CH4/6 CM/S    COMP   COMP›); SPACE(16);
        PRINTTEXT(‹.0          .20        .40         .60          .80          .100     CM/S›); NLCR
END KOP;

PROG:   CARRIAGE(6); PRINTTEXT(‹PRAG-DATA-DECODER-MULTIPURPOSE›); SPACE(75); PRINTTEXT(‹VERSIE:›);  ABSFIXT(10,0,VERSIE);
        CARRIAGE(6); TYD[1]:=TIME;
        BEGIN   INTEGER ARRAY ADMIN[1:25]; REAL ARRAY YK[1:25],AFW[0:8]; INTEGER J;
                FOR J:=1 STEP 1 UNTIL 8 DO  ADMIN[J]:=READ;

                METNO:=ADMIN[8];
                PRINTTEXT(‹K.N.M.I.      DE BILT      NETHERLANDS›):      CARRIAGE(6);
                PRINTTEXT(‹          CAMPAIGN          STATION/     INSTRUMENT       WATER          INSTRUMENT    INSTRUMENT›);
                 NLCR;
                PRINTTEXT(‹
                 NLCR; NLCR;                             PERIOD       DEPTH(M)        DEPTH(M)          TYPE           NUMBER›);
                FOR J:= 1,2,5,6,7,8 DO ABSFIXT(12,0,ADMIN[J]);      CARRIAGE(4):
TESTTYPE:       TYPE:=ADMIN[7]:
                IF TYPE=21 ∨ TYPE=22 THEN AANDERAA:=TRUE ELSE IF TYPE=11 ∨ TYPE=12 THEN AANDERAA:=FALSE
                ELSE BREEKAF(‹FOUT TYPE-NUMMER›):
                BM:=READ;  EM:=READ:
                ADMIN[22]:=ADMIN[12]:=DAG:=ENTIER(BM*⏨-4);
                YK[23]:=YK[21]:=TIJD:=BM-DAG*⏨+4; ADMIN[21]:=  S:=ENTIER(TIJD/100);
                ADMIN[13]:=ENTIER(EM*⏨-4);    YK[22]:=EM-ADMIN[13]*⏨+4;
                V:=ADMIN[11]:=READ;    N:=60/V;
                YK[1]:=READ;    YK[2]:=READ;
                IF  AANDERAA THEN BEGIN FOR J:= 11,12,13,14 DO YK[J]:=READ;  END:
                YK[3]:=READ;    YK[4]:=READ;
                FOR J:= 0 STEP 1 UNTIL 8 DO  AFW[J]:=READ;
                YK[5]:=READ;  YK[6]:=READ;
                ADMIN[16]:=READ;    JUMP:=1023+ADMIN[16];
                RHULP:=READ; ADMIN[9]:=WERK:=RHULP;
                IF WERK < 0 ∨ WERK > 9  ∨ RHULP‡WERK THEN BREEKAF(‹FOUT IN VOORLOOPBAND›);
                IF WERK=3 THEN ADMIN[9]:=0;

                SPACE(13);
                PRINTTEXT(‹DEGR  MIN       DEGR  MIN›); NLCR;
                PRINTTEXT(‹POSITION:›);
                H:=ABS(ADMIN[3]); ABSFIXT(6,0,H÷100);  ABSFIXT(4,0,H-H÷100*100);
                IF ADMIN[3] > 0 THEN PRINTTEXT(‹N›) ELSE PRINTTEXT(‹S›);
                H:=ABS(ADMIN[4]):  ABSFIXT(6,0,H÷100);  ABSFIXT(4,0,H-H÷100*100):
                IF ADMIN[4] > 0 THEN PRINTTEXT(‹E›) ELSE PRINTTEXT(‹W›);
                CARRIAGE(4);
                PRINTTEXT(‹TIME OF FIRST MEASUREMENT GMT:›);  ABSFIXT(10,0,ADMIN[12]):  ABSFIXT(6,0,YK[21]); NLCR;
                PRINTTEXT(‹TIME OF  LAST MEASUREMENT GMT:›);  ABSFIXT(10,0,ADMIN[13]):  ABSFIXT(6,0,YK[22]); CARRIAGE(4);
                PRINTTEXT(‹TIME INTERVAL IN MINUTES:›): ABSFIXT(2,0,V);  CARRIAGE(4):
```

```
                OUTARRAY(DRUM,0,ADMIN): OUTARRAY(DRUM,40,YK): OUTARRAY(DRUM,90,AFW);
        END TRANSFERBLOK;

        OPTIONS;

EINDE ADMINISTRATIE:    TYD[2]:=TIME:  ADRES:=120000:
SKIP:           J:=REHEP;  IF J=0∨J=11∨J=128∨J=64 THEN GOTO SKIP:    WAG[1]:=J;
                FOR I:= 2 STEP 1 UNTIL 200 DO
                BEGIN WAG[I]:=REHEP; IF WAG[I]=255 THEN BEGIN OUTARRAY(DRUM,ADRES,WAG): GOTO CC: END: END:
                OUTARRAY(DRUM,ADRES,WAG): ADRES:=ADRES+200;
LAB:            FOR I:= 1 STEP 1 UNTIL 200 DO
                BEGIN WIG[I]:=REHEP; IF WIG[I]=255 THEN BEGIN OUTARRAY(DRUM,ADRES,WIG): GOTO CC; END; END:
                OUTARRAY(DRUM,ADRES,WIG); ADRES:=ADRES+200;
                HOLD(WAG);
                FOR I:= 1 STEP 1 UNTIL 200 DO
                BEGIN WAG[I]:=REHEP; IF WAG[I]= 255 THEN BEGIN OUTARRAY(DRUM,ADRES,WAG); GOTO CC; END:END:
                OUTARRAY(DRUM,ADRES,WAG); ADRES:=ADRES+200;
                HOLD(WIG);
                GOTO LAB;

CC:             PERF:=SOMPAR:=TW:=0;
                TYD[3]:=TIME;
                ADRES:=120000; HOLD(WIG); HOLD(WAG);
                INARRAY(DRUM,ADRES,WIG);
                LOOP:=1; BLIST:=FALSE;
                ADRES:=ADRES+200;
                INARRAY(DRUM,ADRES,WAG);
                HOLD(WIG);
                ADRES:=ADRES+200:
                COMMENT NU EERSTE REFERENTIEPULS OPZOEKEN; REGEL:=H:=0;

BEGIN   COMMENT KRAAK-BLOK   DECODEERT DE CHARACTER-LIST, TEST OP DE JUISTE STRUCTUUR, VOEGT DE GEDECODEERDE
                GETALLEN AAN EEN DRUM-FILE TOE, BOUWT IN GEVAL VAN EEN AANDERAA METER EEN FILE VAN
                ZOUT- EN DIEPTE-GEGEVENS OP.
                IN GEVAL VAN INCOMPLETE OF OVERCOMPLETE WAARNEMINGEN WORDT EEN ADEQUAAT AANTAL
                'NUL-WAARNEMINGEN' TOEGEVOEGD.
                RICHTING EN SNELHEIDS-GETALLEN WORDEN GETEST OP HET VOORKOMEN VAN EEN GAP,DIT WORDT
                IN DE DRUMFILE AANGETEKEND.
                PLESSEY-SNELHEIDSGETALLEN WORDEN VAN AFTREKKEN OMGEZET NAAR OPTELLEN OM EEN GELIJK
                GEDRAG ALS DE AANDERAA IN DE VERWERKING TE KRIJGEN:

        INTEGER I,J,GETAL,TMEAN,VTMEAN,HTEL,VRG,VSG,K,DELTA,VDELTA,II,IHULP,VIEL,DTEL,TEST,
                DRIE,DELTR,GAPTEL,ENDTEL,T,TTWEE:
        INTEGER ARRAY MEET[1:6],BT,AT[1:25],ASD,BSD[1:10];
        BOOLEAN TEMPTEST,BDUMP:

        BOOLEAN PROCEDURE GAP(IHULP); VALUE IHULP; INTEGER IHULP:
                GAP:= IHULP=0 ∨ IHULP=1 ∨ IHULP=1023;

        PROCEDURE DUMPMEET:
        BEGIN   COMMENT COPIEERT HET MEET-ARRAY IN EEN VAN DE BUFFERS AT/ASD  OF BT/BSD.
                        VULT EEN 1 IN ALS DE METING INCOMPLEET IS (TE DETECTEREN ALS FASE=FALSE).
                        VERZORGT DE OMZETTING VAN PLESSEY NAAR AANDERAA-GEDRAG.
                        TEST OP HET VOORKOMEN VAN EEN GAP.
                        HOUDT DE ADMINISTRATIE VAN DE BUFFERS BIJ EN TRANSPORTEERT EEN VOLLE BUFFER
                        NAAR DE TROMMEL TE BEGINNEN BIJ ADRES 200 VOOR DE SNELHEIDS-RICHTING FILE
                        EN BIJ ADRES 80000 VOOR DE SD-FILE:

                IF BDUMP THEN GOTO BUFFER1;
BUFFER2:                BT[5*I+5]:= IF ¬FASE THEN 1 ELSE  ( IF GAP(MEET[3+K]) THEN 10 ELSE 0 )
```

```
                                            +( IF GAP(MEET[4+K]) THEN 100 ELSE 0):
                     IF AANDERAA THEN BEGIN  BSD[2*I+1]:=MEET[3];  MEET[3]:=MEET[5];
                                             BSD[2*I+2]:=MEET[4];  MEET[4]:=MEET[6];
                                       END
                                  ELSE BEGIN IF FASE THEN MEET[4]:=1023-MEET[4]; END;
                     FOR J:=1 STEP 1 UNTIL 4 DO BT[5*I+J]:=MEET[J];
                     I:=I+1;
                     IF I < 5 THEN GOTO DUMPKLAAR;

                     OUTARRAY(DRUM,ADR,BT); ADR:=ADR+25;
                     IF AANDERAA THEN BEGIN OUTARRAY(DRUM,SDADR,BSD); SDADR:=SDADR+10;  HOLD(ASD): END:
                     HOLD(AT); I:=0: BDUMP:=TRUE;
                     GOTO DUMPKLAAR:

         BUFFER1:          AT[5*I+5]:= IF ¬FASE THEN 1 ELSE  ( IF GAP(MEET[3+K]) THEN 10 ELSE 0)
                                            +( IF GAP(MEET[4+K]) THEN 100 ELSE 0):
                     IF AANDERAA THEN BEGIN  ASD[2*I+1]:=MEET[3];  MEET[3]:=MEET[5];
                                             ASD[2*I+2]:=MEET[4];  MEET[4]:=MEET[6];
                                       END
                                 ELSE IF FASE THEN MEET[4]:=1023-MEET[4];
                     FOR J:= 1 STEP 1 UNTIL 4 DO AT[5*I+J]:=MEET[J];
                     I:=I+1:
                     IF I < 5 THEN GOTO DUMPKLAAR;

                     OUTARRAY(DRUM,ADR,AT); ADR:= ADR+25;
                     IF AANDERAA THEN BEGIN  OUTARRAY(DRUM,SDADR,ASD); SDADR:=SDADR+10; HOLD(BSD): END;
                     HOLD(BT); I:=0:  BDUMP:=FALSE;
                DUMPKLAAR:  IF ADR > 119999 THEN BREEKAF({ADRESFOUT IN DUMPMEET});
                END DUMPMEET;

         INTEGER PROCEDURE LIST:
         BEGIN   COMMENT LEVERT HET SYMBOOL AAN DE KOP VAN EEN LIJST SYMBOLEN AF.
                         ZET DE KOPWIJZER EEN STAP VERDER, HOUDT BIJ OF EEN VAN DE
                         TWEE GEBRUIKTE BUFFERS LEEG IS EN START DAN HET VULLEN
                         EN WACHT TOT DE TWEEDE BUFFER VOL IS EN ZET DE KOPWIJZER
                         OP DE EERSTE POSITIE VAN DIE TWEEDE BUFFER.
                         LEVERT VOORTS DE PARITEIT IN BOOLEAN FOUTPAR AF;
                 IF BLIST THEN GOTO TWEE;
                 LIST:=WIG[LOOP]; FOUTPAR:=PAR[WIG[LOOP]];
                 IF LOOP=200 THEN
                 BEGIN   INARRAY(DRUM,ADRES,WIG);
                         ADRES:=ADRES+200:
                         BLIST:=TRUE;
                         LOOP:=1;
                         HOLD(WAG);
                         GOTO ENDLIST;
                 END;
                 LOOP:=LOOP+1:
                 GOTO ENDLIST;
         TWEE:   LIST:=WAG[LOOP]; FOUTPAR:=PAR[WAG[LOOP]];
                 IF LOOP=200 THEN
                 BEGIN   INARRAY(DRUM,ADRES,WAG);
                         ADRES:=ADRES+200:
                         BLIST:=FALSE:
                         LOOP:=1;
                         HOLD(WIG);
                         GOTO ENDLIST;
                 END:
                 LOOP:=LOOP+1;
         ENDLIST:
```

```
         END:

                     INTEGER ARRAY VERTAAL[0:255],HULPMEET[1:9];
                     BOOLEAN FOUTPAR,SKIPMEET;
                     COMMENT NU WORDT HET VERTAAL-ARRAY GEVULD, AFHANKELIJK VAN HET TYPE STROOMMETER.
                             EEN REFERENTIE-PULS IS TE HERKENNEN AAN EEN NEGATIEF ELEMENT IN VERTAAL,
                             TRUE-ZERO-PONSINGEN GECOMBINEERD MET EEN NUMERIEKE PONSING AAN HET FEIT,
                             DAT HET ELEMENT IN ABSOLUTE WAARDE 50 IS.
                             EEN BLANK (ELEMENT 0) WORDT WEERGEGEVEN MET -255 EN EEN 8-GATS ERASE MET 255;
                     FOR H:=0 STEP 1 UNTIL 31 DO VERTAAL[H]:=VERTAAL[64+H]:=VERTAAL[128+H]:=VERTAAL[192+H]:=H;
                     FOR H:=1 STEP 1 UNTIL 31 DO VERTAAL[32+H]:=VERTAAL[96+H]:=VERTAAL[160+H]:=VERTAAL[224+H]:=50;
                     FOR H:=32,96,160,224 DO VERTAAL[H]:=0;

                     IF TYPE=12 ∨ TYPE=22 THEN BEGIN FOR H:=128 STEP 1 UNTIL 254 DO VERTAAL[H]:=-VERTAAL[H]: END
                                          ELSE BEGIN FOR H:=64 STEP 1 UNTIL 127 DO BEGIN VERTAAL[H]:=-VERTAAL[H];
                                                                                       VERTAAL[128+H]:=-VERTAAL[128+H];
                                                                                  END;
                                               END;
                     VERTAAL[0]:=-255; VERTAAL[255]:=255;

                     NEWPAGE; H:=0;
         REFNEXT:    CHAR:=VERTAAL[LIST]; IF CHAR=-255 THEN GOTO REFNEXT;
                     IF CHAR=255 THEN BREEKAF({ END-CHARACTER FOUND BEFORE FIRST REFERENCE});
                     IF CHAR > 0 THEN BEGIN H:=H+1; GOTO REFNEXT; END:
                     IF H ≠ 0 THEN
                     BEGIN   PRINTTEXT({ CHARACTERS BEFORE PEFERENCE: });ABSFIXT(4,0,H);NLCR;NLCR: REGEL:=3 END:
                     FASE:=TRUE:

                     TEMPTEST:= WERK < 2 ;  BDUMP:= FALSE;
                     ADR:=200:  SDADR:=80000,
                     TOTAL:=I:=HTEL:=TMEAN:=VTMEAN:=TTWEE:=VERTAAL:=TZTEL:=0;
                     DRIE:=T:=1;
                     IF AANDERAA THEN K:=2  ELSE K:=0:
                     TEST:=9+2*K;

                     SKIPMEET:=FALSE;  II:=0;

 CHECKEND:           IF CHAR=255 THEN BEGIN  IF T ≥ TEST THEN BEGIN COMMENT DE LAATSTE INHOUD VAN MEET WORDT NOG
                                                                            WEGGESCHREVEN;
                                                                    TW:=TW+1;
                                                                    DUMPMEET;
                                                               END;
                                          TWBR[13]:=TOTAL:=TW;
                                          GOTO ENDMARK;
                                     END CHAR=255;
 TESTREF:            IF CHAR < 0 THEN BEGIN  CHAR:=ABS(CHAR); SKIPMEET:=FALSE;
                                 IF CHAR = 50 THEN BEGIN TZTEL:=TZTEL+1;
                                                         PRINTTEXT({ONJUISTE TZ-PONSING IN REFERENTIE BIJ TW=});
                                                         ABSFIXT(6,0,TW+2): NLCR;
                                                         SKIPMEET:=TRUE:
                                                    END:
                                 IF FOUTPAR THEN BEGIN SOMPAR:=SOMPAR+1;
                                                         PRINTTEXT({FOUTE PARITEIT IN REFERENTIE BIJ TW=});
                                                         ABSFIXT(6,0,TW+2); NLCR;
                                                         SKIPMEET:=TRUE;
                                                 END;
                                 IF FASE THEN BEGIN  IF T = 1    THEN BEGIN GETAL:=CHAR;GOTO NEXTCHAR; END
                                                                 ELSE IF T = TEST THEN GOTO NORMAL
                                                                 ELSE GOTO REFASE:
                                                 END
```

```
                                                                      ELSE IF T=TEST THEN BEGIN FASE:=TRUE; GOTO NORMAL: END
                                                                                               ELSE GOTO REFASE;
                                                 END;
TESTPAR:                              IF FOUTPAR THEN BEGIN   SOMPAR:=SOMPAR+1; SK PMEET:=TRUE;
                                                         PRINTTEXT(‹FOUTE PARITEIT IN METING, KANAAL:›);
                                                         ABSFIXT(6,0,TW+1); ABSFIXT(4,0,II+1); NLCR;
                                                 END;

TESTTZ:                         IF CHAR = 59 THEN BEGIN TZTEL:=TZTEL+1; SKIPMEET:=TRUE;
                                                                              PRINTTEXT(‹ONJUISTE TZ-PONSING IN METING, KANAAL:›);
                                                                              ABSFIXT(6,0,TW+1); ABSFIXT(4,0,II+1); NLCR;
                                                             END:
                                      IF T-T÷2*2 ≠ 0 THEN BEGIN              IF ¬ SKIPMEET THEN GETAL:=CHAR;
                                                                              GOTO NEXTCHAR;
                                                               END;
                                      II:=II+1; IF II > 6 THEN GOTO NEXTCHAR;
                                      IF SKIPMEET THEN BEGIN  SKIPMEET:=FALSE; IF II > 2+K THEN MEET[II]:=0: END
                                                   ELSE MEET[II]:=32*GETAL+CHAR;
NEXTCHAR:                             CHAR:=VERTAAL[LIST]; IF CHAR=-255 THEN GOTO NEXTCHAR; T:=T+1; GOTO CHECKEND;
NORMAL:                         IF TTWEE ≠ 0 THEN BEGIN PRINTTEXT(‹(NU GOED) ONVOLLEDIGE METING:›);
                                                                              ABSFIXT(6,0,TW); NLCR; PERF:=PERF+1;
                                                                              FASE:=FALSE; DUMPMEET; FASE:=TRUE;
                                                                              TTWEE:=0;
                                                             END;
                                      TW:=TW+1; DUMPMEET; GETAL:=CHAR;
                                      IF TEMPTEST THEN BEGIN  HTEL:=HTEL+1; TMEAN:=TMEAN+MEET[2];
                                                                 IF HTEL=N THEN BEGIN    TMEAN:=TMEAN/N; HTEL:=0;
                                                                                         IF ABS(TMEAN-VTMEAN) < 3 THEN
                                                                                 BEGIN   DRIE:=DRIE+1;
                                                                                         IF DRIE=3 THEN BEGIN    TEMPTEST:=FALSE;
                                                                                                                 PRINTTEXT(‹START TEMP:›);
                                                                                                                 ABSFIXT(4,0,VTMEAN);
                                                                                                                 ABSFIXT(6,0,TW); NLCR;
                                                                                                                 GOTO RESET:
                                                                                                         END;
                                                                                 END


                                                                                                    ELSE DRIE:=1;
                                                                                         VTMEAN:=TMEAN;
                                                                                         TMEAN:=0;
                                                                                 END;
                                                            END:
RESET:                                T:=1; II:=0; GOTO NEXTCHAR;
REFASE:                         IF T+TTWEE=TEST THEN GOTO VERTAALFOUT;
                                      FASE:=FALSE;
                                      IF T < TEST THEN GOTO STOREHULP;
                                      IF TTWEE ≠ 0 THEN BEGIN  DUMPMEET; PERF:=PERF+1;
                                                                              PRINTTEXT(‹(NU OOK FOUT) ONVOLLEDIGE METING:›); ABSFIXT(6,0,TW):
                                                                              NLCR; END;
                                       TWEE:=0; GETAL:=CHAR;
ONVOLLEDIG:                           TW:=TW+1; T:=T+1-TEST; DUMPMEET; PRINTTEXT(‹ONVOLLEDIGE METING:›); ABSFIXT(6,0,TW); NLCR;
                                      PERF:=PERF+1: IF T > 1 THEN GOTO ONVOLLEDIG;
                                      GOTO RESET:
VERTAALFOUT:                    VERTAAL:=VERTAAL+1: FASE:=TRUE; TTWEE:=0;
                                II:=HULPMEET[7]:
                                IF HULPMEET[8] = 0 THEN
                                      BEGIN   IHULP:=1;
EVEN:                                         HULPMEET[II+IHULP]:=32*REMAINDER(MEET[IHULP],32)+MEET[IHULP+1]÷32;
                                              IHULP:=IHULP+1:
                                              IF IHULP+II ≤ 3+K THEN GOTO EVEN;
```

```
                                              HULPMEET[4+K]:=32*REMAINDER(MEET[IHULP],32)+GETAL;
                                      END
                                      ELSE
                                      BEGIN   IHULP:=1;
ONEVEN:                               HULPMEET[II+IHULP]:=MEET[IHULP];
                                              IHULP:=IHULP+1:
                                              IF IHULP+II ≤ 4+K THEN GOTO ONEVEN:
                                              IF HULPMEET[9] = 1 ∧ II > 1+K THEN HULPMEET[II+1]:=0;
                                      END:
                                GETAL:=CHAR;
                                FOR IHULP:=1 STEP 1 UNTIL 6 DO MEET[IHULP]:=HULPMEET[IHULP];
                                DUMPMEET:
                                PRINTTEXT(‹VERTAALFOUT IN METING: ›); ABSFIXT(6,0,TW); NLCR;
                                GOTO RESET;
STOREHULP:                      TW:=TW+1: TTWEE:=T-1: HULPMEET[9]:=0;
                                FOR IHULP:= 1 STEP 1 UNTIL 6 DO HULPMEET[IHULP]:=MEET[IHULP];
                                IF T-T÷2*2 = 0 THEN BEGIN         HULPMEET[7]:=II+1; HULPMEET[8]:=0;
                                                                  IF ¬SKIPMEET THEN HULPMEET[II+1]:=32*GETAL+CHAR;
                                                    END
                                                ELSE
                                                    BEGIN         HULPMEET[7]:=II; HULPMEET[8]:=1:
                                                                  IF SKIPMEET THEN HULPMEET[9]:=1;
                                                    END;
                                GETAL:=CHAR;
                                GOTO RESET;

ENDMARK:                        MEET[1]:=999999; FOR IHULP:=I STEP 1 UNTIL 4 DO DUMPMEET:
                                TYD[4]:=TYD[5]:=TIME:   MEANT:=VTMEAN;
                                END KRAAK-DECODEERBLOK;
                                TWBR[12]:=TWFIRST:=1; FOR J:=1 STEP 1 UNTIL 11 DO TWBR[J]:=TOTAL:
                                IF WERK > 2 ∧ WERK ≠ 9 THEN GOTO DUMP;
                                BEGIN   COMMENT IN DIT BLOK WORDEN DE VERSCHILLENDE TESTS UITGEVOERD.
                                                HET IS HET DEFINITIEGEBIED VAN DE DAARBIJ GEBRUIKTE BUFFERS
                                                EN LEESPROCEDURES;
                                        BOOLEAN BLOOP:
                                        INTEGER I,J,LOOP,NRNEXT,ADR,IHULP,REGEL:
                                        INTEGER ARRAY A,B,C[1:5],WIG,WAG[1:200]:

                                        PROCEDURE INITNEXT(NUMMER); VALUE NUMMER: INTEGER NUMMER;
                                        BEGIN   COMMENT MAAKT ALLES GEREED VOOR DE EERSTE AANROEP VAN NEXT, DIE DAN
                                                        ALS EERSTE DE MET'NUMMER' GEGEVEN METING LEVERT;
                                                NRNEXT:=NUMMER-1;
                                                ADR:=200+5*NUMMER-5; IF ADR > 119999 THEN BREEKAF(‹ADRESFOUT IN INITNEXT›);
                                                INARRAY(DRUM,ADR,WIG);  ADR:=ADR+200:
                                                INARRAY(DRUM,ADR,WAG);  ADR:=ADR+200:
                                                LOOP:=0;
                                                BLOOP:=FALSE:
                                                HOLD(WIG):
                                        END INITNEXT;

                                        PROCEDURE NEXT:
                                        BEGIN   COMMENT IN A WORDT DE VOLGENDE METING AFGEGEVEN ZOALS DIE VOLGT UIT DE
                                                        BUFFERS WIG EN WAG DIE VAN AF DE TROMMEL GEVULD WORDEN:
                                                NRNEXT:=NRNEXT+1; IF ADR > 119999 THEN BREEKAF(‹ADRESFOUT IN NEXT›);
                                                IF BLOOP THEN GOTO BUFFERWAG:
                          BUFFERWIG:            FOR J:= 1 STEP 1 UNTIL 5 DO A[J]:=WIG[LOOP+J];
                                                LOOP:=LOOP+5;
                                                IF LOOP < 200 THEN GOTO ENDNEXT:
                                                INARRAY(DRUM,ADR,WIG); ADR:=ADR+200:
                                                LOOP:=0;  BLOOP:=TRUE;
```

```
                        HOLD(WAG);
                        GOTO ENDNEXT;
        BUFFERWAG:      FOR J:= 1 STEP 1 UNTIL 5 DO A[J]:=WAG[LOOP+J];
                        LOOP:=LOOP+5;
                        IF LOOP < 200 THEN GOTO ENDNEXT;
                        INARRAY(DRUM,ADR,WAG);  ADR:=ADR+200;
                        LOOP:=0;  BLOOP:=FALSE;
                        HOLD(WAG);
        ENDNEXT:        END NEXT;

                IF WERK=2  THEN GOTO TESTSTROOM;
                BEGIN   COMMENT IN DIT BLOK WORDEN DE EERSTE EN DE LAATSTE METING ONDER WATER GEZOCHT
                                AAN DE HAND VAN RESP. HET GEVONDEN EERSTE UURGEMIDDELDE EN DE
                                TEMPERATUURSPRONG AAN HET EINDE VAN DE REEKS;
                        INTEGER DRIE,TMEAN,VTMEAN,GETAL,HTEL,P;
                        IF SOMPAR+TZTEL+PERF+VERTAAL ≠ 0 THEN NEWPAGE;
                        IF WERK = 9 THEN BEGIN INITNEXT(1); GOTO PRINTNEXT; END;
        INITIALIZE:     DRIE:=0;  INITNEXT(1);
        FINDFIRST:      NEXT;
                        IF A[1]=999999 THEN BEGIN PRINTTEXT(‹TEMP. TEST UNSUCCESFUL›); NLCR; GOTO TESTSTROOM; END;
                        IF ABS(A[2]-MEANT) < 3 THEN DRIE:=DRIE+1
                                                ELSE DRIE:=0;
                        IF DRIE ≠ 3  THEN GOTO FINDFIRST;
                        TWFIRST:=NRNEXT-2;
                        INITNEXT(1); ABSFIXT(8,0,DAG); NLCR; KOP; REGEL:=REGEL+3;
        PRINTFIRST:     NEXT;
                        ABSFIXT(3,0,NRNEXT);  ABSFIXT(4,0,A[1]);
                        IF ABS(A[1]-METNO) < 10 THEN SPACE(1) ELSE PRINTTEXT(‹R›);
                        ABSFIXT(4,0,A[2]);  SPACE(7);
                        ABSFIXT(4,0,A[3]);  SPACE(6);
                        ABSFIXT(4,0,A[4]);  NLCR;
                        REGEL:=REGEL+1;
                        IF A[5]=0 ∧ NRNEXT ≥ TWFIRST THEN GOTO FINDLAST;
                        TIJD:=TIJD+V; TIMING(TIJD);
                        IF (ENTIER(TIJD/100)>S∧S≠0)∨(TIJD/100<1∧S=23)∨(S=0∧TIJD>60 ) THEN
                        BEGIN   S:=S+1; IF S=24 THEN BEGIN S:=0; DAGTEL; END; END;
                        IF S=0 THEN
                        BEGIN   IF REGEL>58 THEN BEGIN NEWPAGE; ABSFIXT(8,0,DAG); NLCR; KOP; REGEL:=3; END
                                ELSE BEGIN ABSFIXT(8,0,DAG); REGEL:=REGEL+1; NLCR; END;
                        END
                        ELSE IF REGEL= 60 THEN BEGIN NEWPAGE; ABSFIXT(8,0,DAG); NLCR; KOP; REGEL:=3; END;

                        GOTO PRINTFIRST;
        FINDLAST:       TWFIRST:=NRNEXT;
                        HTEL:=DRIE:=TMEAN:=0;
                        VTMEAN:=MEANT; NEWPAGE;

        TESTMEANT:      NEXT;
                        IF A[1] = 999999 THEN GOTO TESTTIME;
                        IF ABS(VTMEAN-A[2]) > 10 THEN  BEGIN  DRIE:=DRIE+1;  IF DRIE=3 THEN GOTO PRINTLAST; END
                                                       ELSE BEGIN  DRIE:=0;  HTEL:=HTEL+1; TMEAN:=TMEAN+A[2]; END;
                        IF HTEL=N THEN BEGIN VTMEAN:=TMEAN/N;  TMEAN:=HTEL:=0;   END ;
                        GOTO TESTMEANT;

        PRINTLAST:      PRINTTEXT(‹SLOT VAN DE MEETREEKS›); NLCR; NLCR;
                        TOTAL:=NRNEXT-3; INITNEXT(TOTAL+1);   A[1]:=999999; OUTARRAY(DRUM,200+5*TOTAL,A); HOLD(A);
                        FOR P:= 1 STEP 1 UNTIL 11 DO TWBR[P]:=TOTAL; REGEL:=3;
        PRINTNEXT:      NEXT;
                        IF A[1]=999999  THEN GOTO TESTTIME;
                        ABSFIXT(6,0,NRNEXT);
```

```
                        FOR P:=1 STEP 1 UNTIL 4 DO ABSFIXT(4,0,A[P]); IF A[5]=1 THEN PRINTTEXT(‹A›);
                        NLCR;
                        GOTO PRINTNEXT;
        TESTTIME:       IF WERK = 9 THEN GOTO STATISTIEK;
                        CARRIAGE(4);  PRINTTEXT(‹TEMP.TEST TOOK (SEC):›); ABSFIXT(6,0,TIME-TYD[4]); CARRIAGE(4);
                END TEMPERATUUR-BLOK;

TESTSTROOM:     BEGIN   COMMENT IN DIT BLOK WORDEN DE RICHTINGS- EN SNELHEIDSGETALLEN GETEST.
                                VERDER WORDT GETEST OP MOGELIJKE KLOKFOUTEN;

                        INTEGER P,Q,VRG,VSG,VDELTA,IHULP,VTEL,DTEL,DELTR,GAPTEL,ENDTEL,T,K,EIND,LNR,LSG,
                                VNR,BNR,CNR,GAPMARGE,PLUSTEL;
                        BOOLEAN MIN;

                        BOOLEAN PROCEDURE GAP(IHULP);  VALUE IHULP; INTEGER IHULP;
                                GAP:= IHULP=0 ∨ IHULP=1 ∨ IHULP=1023;

        BOOLEAN PROCEDURE FOUT(VSG,GETAL,VTEL);
        VALUE VTEL; INTEGER VSG,GETAL,VTEL;
        BEGIN   INTEGER DELTA,HULP;
                DELTA:=GETAL-VSG; IF DELTA<0 THEN
                BEGIN   MIN:=TRUE; IF VSG>(IF AANDERAA THEN 700 ELSE 800) ∧ GETAL<(IF AANDERAA THEN 300 ELSE 200)
                        THEN DELTA:=DELTA+JUMP ELSE GOTO POTMETER;
                END ELSE IF MIN THEN
                BEGIN   IF PLUSTEL>2 THEN GOTO POTMETER;
                        PLUSTEL:=0; MIN:=FALSE;
                END;
                DELTA:=DELTA/(1+VTEL); HULP:=ABS(DELTA-VDELTA);
                IF (HULP>20+GAPMARGE+K*5)∧(HULP>0.2*VDELTA+GAPMARGE) THENFOUT:=TRUE ELSE
                BEGIN   FOUT:=FALSE; VDELTA:=DELTA; END;
                GOTO EIND;
POTMETER:       PLUSTEL:=PLUSTEL+1; FOUT:=TRUE; IF PLUSTEL=12 THEN
                BEGIN   CARRIAGE(4); PRINTTEXT(‹SNELHEIDSPOTMETER LOOPT VERKEERD OM EINDE›);
                        ABSFIXT(10,0,NRNEXT);CARRIAGE(4);
                        TOTAL:=NRNEXT-12; FOR J:=TEND STEP 1 UNTIL 11 DO TWBR[J]:=TOTAL;
                        A[1]:=999999; OUTARRAY(DRUM,200+5*TOTAL,A); HOLD(A); GOTO TESTRICHT;
                END;
EIND:           END FOUT;

        INTEGER PROCEDURE DEL(VSG,GETAL,VTEL); VALUE VTEL; INTEGER VSG,GETAL,VTEL;
        BEGIN   INTEGER HULP;
                HULP:=GETAL-VSG; IF HULP<(IF AANDERAA THEN -750 ELSE -850) THEN HULP:=HULP+JUMP;
                DEL:=HULP/VTEL;
        END DEL;

        PROCEDURE POTPRINT;
        BEGIN   INTEGER L;
                FOR L:=(IF TWBR[12]=1 THEN 1 ELSE TWBR[12]+1),L+1 WHILE L≤TWFIRST DO
                BEGIN   ABSFIXT(3,0,L); INARRAY(DRUM,200+5*L-5,C); HOLD(C);
                        ABSFIXT(4,0,C[1]); IF ABS(C[1]-METNO)<10 THEN SPACE(1) ELSE PRINTTEXT(‹R›);
                        ABSFIXT(4,0,C[2]); SPACE(7); ABSFIXT(4,0,C[3]); SPACE(6); ABSFIXT(4,0,C[4]);
                        IF C[5]=1 THEN PRINTTEXT(‹A›); NLCR;
                        TIJD:=TIJD+V; TIMING(TIJD);
                        IF (ENTIER(TIJD/100)>S∧S≠0)∨(TIJD/100<1∧S=23)∨(S=0∧TIJD>60)
                        THEN    BEGIN   S:=S+1; IF S=24 THEN BEGIN S:=0; DAGTEL; END; END;
                        IF S=0 THEN BEGIN IF REGEL>58 THEN BEGIN NEWPAGE; ABSFIXT(8,0,DAG);NLCR;KOP;REGEL:=3; END
                                          ELSE BEGIN ABSFIXT(8,0,DAG);NLCR;REGEL:=REGEL+1; END;
                                    END
                        ELSE IF REGEL=60 THEN BEGIN NEWPAGE;ABSFIXT(8,0,DAG);NLCR;KOP;REGEL:=3; END;
                END;
```

```
                    TWBR[12]:=TWFIRST;
               END PUTPRINT;

               BOOLEAN PROCEDURE RFOUT;
               BEGIN   INTEGER HULP;
                       DELTR:=6-ENTIER((VDELTA-K*20)/10);
                       IF DELTR < 0 THEN DELTR:=0;
                       IF DELTR > 6 THEN DELTR:=6;
                       HULP:=100+50*DTEL+20*DELTR;
                       RFOUT:=(ABS(VRG-IHULP)>HULP) ∧ (ABS(VRG-IHULP)<1037-HULP);
               END RFOUT;

                               K:=IF AANDERAA THEN 2 ELSE 0:
                               ENDTEL:=0; TEND:=1: TWBR[12]:=TWFIRST;
STARTTEST:     PRINTTEXT({START STROOMTEST}); NLCR; NLCR; PLUSTEL:=0; MIN:=FALSE;

               INITNEXT(TWFIRST);
               COMMENT ZOEK EERSTE SNELHEIDSGETAL;

FIRST:  NEXT;
               IF A[5] ≠ 0 THEN
               BEGIN   IF A[5] ≠ 1 THEN
                       BEGIN   A[5]:=110; OUTARRAY(DRUM,200 + 5 * NRNEXT-5,A); HOLD(A); END:
                               TWFIRST:=NRNEXT+1; IF TEND=1 THEN PUTPRINT; GOTO FIRST;
               END;
               LNR:=NRNEXT; LSG:=A[4]: GAPTEL:=GAPMARGE:=0;
               COMMENT ZOEK NU TWEEDE SNELHEIDSGETAL;

SECOND: NEXT;
               IF A[5]=1 ∨ A[5]>99 THEN
               BEGIN   GAPTEL:=GAPTEL+1; GOTO SECOND; END;
               VSG:=A[4]; VNR:=NRNEXT: GAPMARGE:=5 * GAPTEL;
               VDELTA:=DEL(LSG,VSG,NRNEXT - LNR); GAPTEL:=0;

               COMMENT ZOEK NU DERDE GETAL EN GA TESTEN:
THIRD:  NEXT;
               IF A[5]=1 ∨ A[5]>99 THEN
               BEGIN   GAPTEL:=GAPTEL+1; GAPMARGE:= 5 * GAPTEL; GOTO THIRD; END:
               IHULP:=A[4];

               IF ¬ FOUT(VSG,IHULP,NRNEXT - VNR - 1) THEN
               BEGIN   NLCR; PRINTTEXT({EERSTE DRIE GOED}); NLCR; NLCR; GOTO UPDATE; END;

               COMMENT 3 PAST NIET BIJ 1 EN 2;

               BNR:=NRNEXT;
               FOR P:=1, P+1 WHILE P<6 DO B[P]:=A[P];

FOURTH: NEXT;
               IF A[5]=1 ∨ A[5]>99 THEN
               BEGIN   GAPTEL:=GAPTEL+1: GAPMARGE:=5 * GAPTEL; GOTO FOURTH; END;
               IHULP:=A[4]:
               IF ¬ FOUT(VSG,IHULP,NRNEXT - VNR - 1) THEN

               BEGIN   COMMENT 4 PAST BIJ 1 EN 2; NLCR;PRINTTEXT({1,2 EN 4 GOED});NLCR;NLCR;
                       B[5]:=B[5] + 100; OUTARRAY(DRUM,200+5*BNR-5,B); HOLD(B);
                       GOTO UPDATE;
               END:
```

```
               COMMENT 4 NOCH 3 PASSEN BIJ 1 EN 2 KIJK OF 1,3 EN 4 WEL BIJ ELKAAR PASSEN;

               VDELTA:=DEL(LSG,B[4],BNR - LNR);

               IF FOUT(B[4],IHULP,NRNEXT - BNR - 1) THEN GOTO TWEEFOUT;

               COMMENT KENNELIJK PAST 2 NIET IN DE REEKS 1,3 EN 4 KEUR 2(=VNR) AF;
               INARRAY(DRUM,200 + 5 * VNR-5,A); HOLD(A): A[5]:=A[5] +100;
               OUTARRAY(DRUM,200 + 5 * VNR - 5,A); HOLD(A);
               NLCR;PRINTTEXT({1,3 EN 4 GOED});NLCR;NLCR;

               VNR:=BNR: VSG:=B[4]; VDELTA:=DEL(VSG,IHULP,NRNEXT - VNR); GOTO UPDATE;

               COMMENT HET VOLGENDE GEDEELTE KOMT IN WERKING ALS 3 NOCH 4 BIJ 1 EN 2 PASSEN EN OOK
               1,3 EN 4 NIET BIJ ELKAAR PASSEN.
               NU  EERST NOG KIJKEN OF 2,3 EN 4 WEL BIJ ELKAAR PASSEN;
TWEEFOUT:      VDELTA:=DEL(VSG,B[4],BNR - VNR);
               IF FOUT(B[4], IHULP,NRNEXT - BNR - 1) THEN
               BEGIN   TWFIRST:=BNR: IF TEND=1 THEN PUTPRINT; GOTO STARTTEST;
                       COMMENT BEGIN OPNIEUW MET 3 ALS EERSTE WAARNEMING;
               END:
               COMMENT 2,3 EN 4 PASSEN BLIJKBAAR WEL. 2 WORDT DE EERSTE;

               INARRAY(DRUM,200 + 5 * LNR - 5,A); HOLD(A); A[5]:=110;
               OUTARRAY(DRUM,200 + 5 * LNR - 5,A); HOLD(A);
               TWFIRST:=VNR; IF TEND=1 THEN PUTPRINT; VNR:=BNR; VSG:=B[4]; VDELTA:=DEL(VSG,IHULP,NRNEXT-VNR);NLCR;
               PRINTTEXT({2,3 EN 4 GOED});NLCR;NLCR: GOTO UPDATE;
      SPEEDLOOP:                 NEXT:
                                 IF A[1]=999999 THEN GOTO TESTRICHT;
                                 IF A[5]≠0 ∧ A[5]≠10 THEN BEGIN GAPTEL:=GAPTEL+1; GAPMARGE:=5*GAPTEL; GOTO SPEEDLOOP; END:
                                 IHULP:=A[4];
                                 IF FOUT(VSG,IHULP,NRNEXT-VNR-1) THEN
                                 BEGIN   BNR:=NRNEXT; PRINTTEXT({FOUT 1});
                                         FOR P:=1 STEP 1 UNTIL 5 DO B[P]:=A[P];
      WHONEXT:                           NEXT;
                                         IF A[1]=999999 THEN BEGIN B[5]:=B[5]+100;
                                                                      OUTARRAY(DRUM,200+5*BNR-5,B): HOLD(B);
                                                                      GOTO TESTRICHT;
                                                              END;
                                         IF A[5]≠0 ∧ A[5]≠10 THEN BEGIN GAPTEL:=GAPTEL+1; GAPMARGE:=5*GAPTEL;GOTO WHONEXT: END;
                                         IHULP:=A[4];
                                         IF FOUT(VSG,IHULP,NRNEXT-VNR-1) THEN GOTO FOUTTWEE;
                                         B[5]:=B[5]+100:
                                         OUTARRAY(DRUM,200+5*BNR-5,B):  HOLD(B); PRINTTEXT({EINDE}); NLCR:
                                 END ENKELE FOUT;

                                 COMMENT NU IS A GOEDGEKEURD EN WORDT DE ADMINISTRATIE VOOR DE VOLGENDE TESTS BIJGEWERKT;
      UPDATE:            LSG:=VSG;  LNR:=VNR;
                                 VSG:=IHULP;  VNR:=NRNEXT;
                                 GAPMARGE:=5*GAPTEL; GAPTEL:=0;
                                 GOTO SPEEDLOOP;

      FOUTTWEE:                  BEGIN   COMMENT IN DIT BLOK WORDT BEKEKEN OF HET AFKEUREN VAN TWEE OPEENVOLGENDE
                                                 SNELHEIDSGETALLEN GERECHTVAARDIGD IS OF DAT ER TOT EEN KLOKSTILSTAND
                                                 BESLOTEN MOET WORDEN;
                                         INTEGER DELTA1,DELTA2,TEL;
                                         INTEGER ARRAY NRFOUT[1:5];
                                         TEL:=0;
```

```
                 COMMENT EERST WORDT NOG BEKEKEN OF ALSNOG AFKEUREN VAN
                          METING VNR HET AFKEUREN VAN BNR EN NRNEXT OVERBODIG MAAKT;
                 DELTA1:=VDELTA;
                 VDELTA:=DEL(LSG,B[4],BNR - LNR);
                 IF ¬ FOUT(B[4],IHULP,NRNEXT-BNR-1) THEN
                 BEGIN    INARRAY(DRUM,200+5*VNR-5,C); HOLD(C);
                          C[5]:=C[5]+100;
                          OUTARRAY(DRUM,200+5*VNR-5,C);
                          VNR:=BNR;  VSG:=B[4];
                          PRINTTEXT(‹ALSNOG     EINDE›); HOLD(C);
                          GOTO UPDATE;
                 END ELSE VDELTA:=DELTA1;
SCHUIF:                   FOR P:= 1 STEP 1 UNTIL 5 DO BEGIN C[P]:=B[P]; B[P]:=A[P]; END;
WHATSNEXT:                         CNR:=BNR; BNR:=NRNEXT; PRINTTEXT(‹FOUT 2›);
                                   NEXT;
                                   IF A[1]=999999 THEN BEGIN B[5]:=B[5]+100;
                                                             OUTARRAY(DRUM,200+5*BNR-5,B);
                                                             C[5]:=C[5]+100;
                                                             OUTARRAY(DRUM,200+5*CNR-5,C);
                                                             IF TEL > 0 THEN BEGIN FOR P:=1 STEP 1 UNTIL TEL DO
                                                                              BEGIN Q:=200+5*NRFOUT[P]-5;
                                                                                    INARRAY(DRUM,Q,A); HOLD(A);
                                                                                    A[5]:=A[5]+100;
                                                                                    OUTARRAY(DRUM,Q,A); HOLD(A);
                                                                              END;
                                                                              END;
                                                             HOLD(C);
                                                             GOTO TESTRICHT;
                                                         END;
                          IF A[5]≠0 ∧ A[5]≠ 10 THEN BEGIN GAPTEL:=GAPTEL+1; GAPMARGE:=5*GAPTEL; GOTO WHATSNEXT; END;
TESTABC:                  DELTA1:=DEL(C[4],B[4], BNR-CNR); DELTA2:=DEL(B[4],A[4],NRNEXT - BNR);
                          IF ABS(DELTA1-DELTA2) < 20+K*5+GAPMARGE ∨
                           ABS(DELTA1-DELTA2) < 0.2*(DELTA1+DELTA2)/2+GAPMARGE THEN
                 BEGIN  COMMENT TESTUITVOER;
                          ABSFIXT(6,0,CNR); ABSFIXT(6,0,BNR); ABSFIXT(6,0,NRNEXT); NLCR;
                          ABSFIXT(6,0,C[4]); ABSFIXT(6,0,B[4]); ABSFIXT(6,0,A[4]);
                          FIXT(8,0,DELTA1); FIXT(6,0,DELTA2); FIXT(6,0,VDELTA); NLCR;
                                 GOTO KLOKFOUT;
                 END;

                                 COMMENT DE DRIE GETALLEN PASSEN WEL BIJ ELKAAR MAAR NIET ZONDER MEER
                                         BIJ HET VOORGAANDE. DAN KAN ER IETS MET DE KLOK GEWEEST ZIJN,
                                         HETGEEN BIJ KLOKFOUT APART ONDERZOCHT WORDT;
BCFOUT:                   IHULP:=A[4];
                                 IF FOUT(VSG,IHULP,NRNEXT-VNR-1) THEN
                                 BEGIN   COMMENT 3 OF MEER NIET BIJELKAAR PASSENDE OPEENVOLGENDE
                                                 SNELHEIDSGETALLEN ZIJN AFGEKEURD;
                                         TEL:=TEL+1;
                                         NRFOUT[TEL]:=CNR;
                                         IF TEL = 3 THEN GOTO ONDERBREKING;
                                         GOTO SCHUIF;
                                         COMMENT GETRACHT WORDT OM 3 OPEENVOLGENDE, BIJEENPASSENDE
                                                 SNELHEIDSGETALLEN TE VINDEN;
                                 END
                                 ELSE
                                 BEGIN COMMENT A BEVAT EEN METING DIE PAST BIJ DE METING'VNR'.
                                               DE TUSSENLIGGENDE METINGEN ZIJN AFGEKEURD;
                                         B[5]:=B[5]+100;
                                         OUTARRAY(DRUM,200+5*BNR-5,B);
```

```
                                         C[5]:=C[5]+100;
                                         OUTARRAY(DRUM,200+5*CNR-5,C);
                                         IF TEL > 0 THEN BEGIN FOR P:=1 STEP 1 UNTIL TEL DO
                                                         BEGIN Q:=200+5*NRFOUT[P]-5;
                                                               INARRAY(DRUM,Q,A); HOLD(A);
                                                               A[5]:=A[5]+100;
                                                               OUTARRAY(DRUM,Q,A); HOLD(A);
                                                         END;
                                                         END;
                                         HOLD(C); PRINTTEXT(‹EINDE›);NLCR;
                                 GOTO UPDATE;
                          END;
KLOKFOUT:        PRINTTEXT(‹KLOKTEST›); NLCR;
                 IHULP:=DEL(VSG,C[4],CNR - VNR);
                                 COMMENT ER ZIJN NU VIER OPEENVOLGENDE SNELHEDEN BEKEND:
                                         VDELTA, IHULP, DELTA1 EN DELTA2.
                                         ER WORDT NU BEKEKEN OF DIE BIJ ELKAAR AANSLUITEN.
                                         WE WETEN REEDS DAT: 1) IHULP PAST NIET BIJ VDELTA
                                                             2) DELTA1 PAST WEL BIJ DELTA2.
                                         DE VRAAG IS NU OF WE MET EEN SNELLE SNELHEIDSVERANDERING TE DOEN
                                         HEBBEN OF MET EEN KLOKFOUT.
                                         EERST WORDT NOG DE ADMINISTRATIE ZOVEEL MOGELIJK BIJGEWERKT ALVORENS DE TEST
                                         WORDT UITGEVOERD;
                                 VDELTA:=DELTA2;
                                 LSG:=B[4];  LNR:=BNR;
                                 GAPMARGE:=5*GAPTEL; GAPTEL:=0;
KLOKTEST:                        VSG:=A[4];  COMMENT VNR KAN ZIJN NIEUWE WAARDE PAS LATER KRIJGEN;
                                 IF ABS(IHULP+DELTA2-2*DELTA1) < 20+K*5+GAPMARGE
                                  ∨ ABS(IHULP+DELTA2-2*DELTA1) < 0.2*DELTA1+GAPMARGE THEN
                                 BEGIN   COMMENT WE HEBBEN KENNELIJK MET EEN SNELLE VERANDERING TE MAKEN;
                                         VNR:=NRNEXT;
                                         GOTO SPEEDLOOP;
                                 END;
                                 COMMENT HET GEHANTEERDE KRITERIUM IS DE HELFT VAN HET VOOR AFZONDERLIJKE
                                         METINGEN GEHANTEERDE.
                                         HIERKOMT HET PROGRAMMA ALS ER KENNELIJK EEN KLOKFOUT IS OPGETREDEN;
ONDERBREKING:                    PRINTTEXT(‹SERIE-VERWERKING ONDERBROKEN BIJ:›);  ABSFIXT(6.0,VNR); NLCR; NLCR;
                                 IF TEND > 1 THEN BEGIN IF VNR-TWBR[TEND-1] < 10 THEN ENDTEL:=ENDTEL+1
                                                                                      ELSE ENDTEL:=0;
                                                  END;
                                 IF TEND < 11 THEN TWBR[TEND]:=VNR;  TEND:=TEND+1;
                                 IF TEND > 10 ∨ ENDTEL > 4 THEN
                                 BEGIN   PRINTTEXT(‹TEVEEL ONZIN›);  NEWPAGE;  REGEL:=0;
                                         TOTAL:=TWBR[TEND-1];
                                         INITNEXT(TOTAL+1);
                                         A[1]:=999999; OUTARRAY(DRUM,200+5*TOTAL,A); HOLD(A);
                                         GOTO PRINTTAPE;
                                 END TEVEEL ONZIN;
MERKKLOK:                        INARRAY(DRUM,200+5*VNR,A);  HOLD(A);
                                 A[1]:=999999;
                                 OUTARRAY(DRUM,200+5*VNR,A);  HOLD(A);
                                 IF TEL = 3 THEN BEGIN COMMENT DE VERWERKING IS ONDERBROKEN OMDAT 5 OPEENVOLGENDE
                                                               METINGEN WERDEN AFGEKEURD. HET ZOEKEN VAN NIEUWE
                                                                    STARTKRITERIA IS NODIG;
                                                         TWFIRST:=VNR+1;
                                                         GOTO STARTTEST;
                                                  END TEL=5;
                                 VNR:=NRNEXT;
                                 GOTO SPEEDLOOP;
                                 COMMENT HET VOLGENDE GEDEELTE WORDT ALLEEN DOORLOPEN ALS ER TEVEEL ONZIN
```

```
                                               IS OPGETREDEN;
        PRINTTAPE:                      NEXT;
                                        IF A[1]=999999 THEN GOTO TESTRICHT;
                                        ABSFIXT(6,0,NRNEXT);
                                        FOR P:= 1 STEP 1 UNTIL 4 DO ABSFIXT(4,0,A[P]);
                                        IF A[5]=1 THEN PRINTTEXT(‹  A›);
                                        NLCR;
                                        GOTO PRINTTAPE;
                                END FOUTTWEE-BLOK;
        TESTRICHT:              BEGIN   PROCEDURE SPEED;
                                        BEGIN   IF A[5]≠1 ∧ A[5]< 100 THEN
                                                 BEGIN  VDELTA:=DEL(VSG,A[4],NRNEXT-VNR); VSG:=A[4];  VNR:=NRNEXT; END;
                                        END     SPEED;
                                        INTEGER LEVEL;

                                        INITNEXT(TWBR[12]); NLCR; PRINTTEXT(‹START RICHTINGSTEST      ›); NLCR; ENDTEL:=1;
        DIRFIRST:                       NEXT;
                                        IF A[5]≠0 THEN GOTO DIRFIRST;
                                        VRG:=A[3]; VNR:=NRNEXT; VSG:=A[4];
                                        GAPTEL:=DTEL:=VDELTA:=0; PRINTTEXT(‹EERSTE GOED›); NLCR;
        DIRECTIONLOOP:                  NEXT;
        TESTEND:                        IF A[1]=999999 THEN
                                        BEGIN   COMMENT HIER KOMT HET PROGRAMMA AAN HET EINDE VAN DE
                                                MEETREEKS OF ALS ER EEN KLOKSTILSTAND GEVONDEN IS;
                                                IF ENDTEL=TEND THEN GOTO VOLG;
                                                ENDTEL:=ENDTEL+1;
                                                IF A[5]≠0 THEN GOTO DIRFIRST;
                                                VRG:=A[3]; VSG:=A[4]; VNR:=NRNEXT;
                                                GAPTEL:=DTEL:=VDELTA:=0;
                                                GOTO DIRECTIONLOOP;
                                        END     SLOT OF KLOKFOUT;
                                        SPEED;
                                        IF A[5]=1 ∨ A[5]=10 ∨ A[5]=110 THEN
                                        BEGIN  DTEL:=DTEL+1;
                                                GAPTEL:=GAPTEL+1;
                                                GOTO DIRECTIONLOOP;
                                        END     GAP OF ONVOLLEDIG;

                                        IHULP:=A[3];
                                        IF RFOUT THEN GOTO SCHUIFB;
        DIRUPDATE:                      VRG:=A[3]; DTEL:=GAPTEL:=0; HOLD(A);
                                        GOTO DIRECTIONLOOP;
        SCHUIFB:                        BNR:=NRNEXT; HOLD(B); PRINTTEXT(‹FOUT 1›);
                                        FOR P:= 1 STEP 1 UNTIL 5 DO B[P]:=A[P];
        NEXTDIR1:                       NEXT; IF A[1]=999999 THEN GOTO TESTEND; SPEED; DTEL:=DTEL+1;
                                        IF A[5] ≠ 0 ∧ A[5] ≠ 100 THEN GOTO NEXTDIR1;
                                        IHULP:=A[3];
                                        IF RFOUT THEN GOTO SCHUIFC;
        TESTBA:                VRG:=B[3]; DTEL:=NRNEXT-BNR-1;
                                        IF RFOUT THEN GOTO BFOUT;
                                        PRINTTEXT(‹EINDE BA›); NLCR;
                                        GOTO DIRUPDATE;
        BFOUT:                          B[5]:=B[5]+10; OUTARRAY(DRUM,195+5*BNR,B);
                                        PRINTTEXT(‹EINDE B›); NLCR; GOTO DIRUPDATE;
        SCHUIFC:                        CNR:=NRNEXT; HOLD(C); PRINTTEXT(‹  FOUT 2›);
                                        FOR P:=1 STEP 1 UNTIL 5 DO C[P]:=A[P];
        NEXTDIR2:                       NEXT; IF A[1]=999999 THEN GOTO TESTEND; SPEED; DTEL:=DTEL+1;
                                        IF A[5] ≠ 0 ∧ A[5] ≠ 100 THEN GOTO NEXTDIR2;
                                        IHULP:=A[3];
```

```
                                       IF RFOUT THEN GOTO TESTDRIE;
        TESTCA:                DTEL:=NRNEXT-CNR-1; VRG:=C[3];
                                       IF RFOUT THEN BEGIN C[5]:=C[5]+10; OUTARRAY(DRUM,195+5*CNR,C); GOTO TESTBA; END;
        TESTBC:                        VRG:=B[3]; IHULP:=C[3]; DTEL:=CNR-BNR-1;
                                       IF RFOUT THEN GOTO BFOUT ELSE BEGIN PRINTTEXT(‹EINDE BC›); NLCR; GOTO DIRUPDATE; END;
        TESTDRIE:                      VRG:=C[3]; DTEL:=NRNEXT-CNR-1;
                                       IF RFOUT THEN BEGIN      C[5]:=C[5]+10; OUTARRAY(DRUM,195+5*CNR,C);
                                                                VRG:=B[3]; DTEL:=NRNEXT-BNR-1;
                                                                IF RFOUT THEN BEGIN  A[5]:=A[5]+10;
                                                                                     OUTARRAY(DRUM,195+5*NRNEXT,A);
                                                                                     GOTO BFOUT;
                                                                              END
                                                                       ELSE BEGIN PRINTTEXT(‹EINDE DRIE›); NLCR;
                                                                                  GOTO DIRUPDATE;
                                                                              END;
                                                      END
                                               ELSE GOTO TESTBC;
                               END RICHTINGSTESTBLOK;
                       END TESTBLOK;
              END BUFFERBLOK;

        VOLG:          TYD[5]:=TIME;
                       BEGIN   COMMENT BEGIN-TIJD INFORMATIE WEGSCHRIJVEN;
                               INTEGER ARRAY SDAG[1:2]; REAL ARRAY KLOK[1:1];
                               SDAG[1]:=S; SDAG[2]:=DAG; OUTARRAY(DRUM,20,SDAG); KLOK[1]:=TIJD; OUTARRAY(DRUM,84,KLOK);
                       END;
                       TWBR[11]:=TOTAL;  TWBR[13]:=TW;
        DUMP:          OUTARRAY(DRUM,25,TWBR);
                       NEWPAGE;
                       IF WERK > 2 THEN GOTO STATISTIEK;

                       BEGIN INTEGER IHULP,DATUM1,DATUM2,EXPTW;
                               REAL HULP;
                               IF TEND=1 THEN
                               BEGIN   PRINTTEXT(‹ER IS GEEN KLOK-STILSTAND GEDETECTEERD VOOR HET EINDE-BAND›); NLCR; NLCR; END;
                               IHULP:=0;
                               DATUM1:=ENTIER(BM*⏨-4); DATUM2:=ENTIER(EM*⏨-4);
                               HULP:=2360-BM+EM+(DATUM1-DATUM2)*⏨+4;
                               IF HULP>2360 THEN IHULP:=1; TIMING(HULP);

                               FOR DAG:= DATUM1+1, DAG+1 WHILE DAG < DATUM2 DO
                               BEGIN   DAGTEL; IHULP:=IHULP+1; END;
                               IHULP:=IHULP*24+ENTIER(HULP/100);
                               IHULP:=IHULP*60+RMOD(HULP,100);
                               EXPTW:=IHULP/V;
                               PRINTTEXT(‹VERWACHT AANTAL WAARNEMINGEN:›);
                               ABSFIXT(6,0,EXPTW); NLCR;
                               IF ABS(EXPTW-TW)/TW < 2*⏨-3 THEN
                               BEGIN PRINTTEXT(‹GESUGGEREERDE TIJDSTAP:›);
                               ABSFIXT(4,4,(V*EXPTW)/TW);
                       END;
                       CARRIAGE(4);
              END TEL-BLOK;
        STATISTIEK:    PRINTTEXT(‹AANTAL WAARNEMINGEN:›);  ABSFIXT(6,0,TW);  NLCR;
                       PRINTTEXT(‹AANTAL PARITEITSFOUTEN:›):   ABSFIXT(4,0,SOMPAR);NLCR;
                       PRINTTEXT(‹AANTAL ONJUISTE TZ-PONSINGEN:›); ABSFIXT(4,0,TZTEL); NLCR;
                       PRINTTEXT(‹AANTAL INCOMPLETE WAARNEMINGEN:›):  ABSFIXT(4,0,PERF); NLCR;
                       PRINTTEXT(‹AANTAL GEVONDEN VERTAALFOUTEN:›); ABSFIXT(4,0,VERTAAL); NLCR;  NLCR;
                       PRINTTEXT(‹         VOORLOOP     WIGWAG       DECODEREN     TESTEN    OVERDRACHT›); NLCR;
                       FOR J := 1 STEP 1 UNTIL 5 DO
```

```
                                        BEGIN ABSFIXT(6,0,TYD(J)); SPACE(4); END;
                              ASKNEXT;

          STOP:    ABSFIXT(6,0,TIME); NLCR;  NLCR: PRINTTEXT(¢EINDE PRAG-DATA-DECODER$);
                   END WERKBLOK:
          END
```

Appendix E.

DATA-COMPUTATION

Algol-60 source listing.

```
740206: BEGIN COMMENT K.N.M.I.-PRAG-DATA-COMPUTATION.
                    ONDERZOEKT DE VIA TROMMEL DOOR DATA-DECODER OVERGEDRAGEN STROOMMETERGEGEVENS
                    OP GECONSTATEERDE FOUTEN. DEZE WORDEN NADER BESCHOUWD EN ZONODIG DOOR GEINTER-
                    POLEERDE GETALLEN VERVANGEN.
                    HET ALDUS VERKREGEN BESTAND WORDT VERWERKT TOT EEN PRINTOUT EN PONSBAND VAN DE
                    (GECORRIGEERDE) MEETWAARDEN ALSMEDE TOT EEN PRINTOUT VAN UURGEMIDDELDEN:
              INTEGER VERSIE;
              VERSIE:=740206;

              BEGIN COMMENT DIT IS HET EIGENLIJKE WERKBLOK;
                    INTEGER METNO,V,ADRES,I,J,N,ADR,TEL,VGT,VGR,VGS,TOTERR,SDADR,DTEL,
                          UURADR,TOTUUR,TOTAL,TEND,K,TW,S,TGW,TWG,TWFIRST,DAG,VDAG,JUMP;

                    REAL A0,A1,B0,B1,C0,C1,KV,KF,APRO,CFS,RAD,SAL,DIEP,RICHT,SNELH,VR,GEMR,TIJD,HSAL,HDIEP;

                    BOOLEAN ONE,PONS,TWO,AANDERAA,COMPL;
                    INTEGER ARRAY A[1:5],TYD[1:6],SD[1:2],TWBR[1:15];
                    REAL ARRAY AFW[0:8],UURW[1:168];

        PROCEDURE BREEKAF(TEXT); STRING TEXT;
        BEGIN   CARRIAGE(4); PRINTTEXT(TEXT); CARRIAGE(4); GOTO STOP; END;

        REAL PROCEDURE SUM(I,A,B,X); VALUE B; INTEGER I,A,B; REAL X;
        BEGIN   REAL S; S:=0;
                FOR I:=A,I+1 WHILE I≤B DO S:=S+X;
                SUM:=S
        END SUM;

        PROCEDURE PONSZUINIG(N,M,X); VALUE N,M,X; INTEGER N,M; REAL X;
        BEGIN   COMMENT PONST GETAL X MET TEKEN, MAXIMAAL N POSITIES VOOR EN MAXIMAAL M POSITIES ACHTER
                        DE DECIMALE PUNT.
                        HET GEDEELTE ACHTER M WORDT AFGEROND.
                        DE DECIMALE PUNT WORDT ALLEEN GEPONST ALS ER NA AFRONDING EEN DECIMALE FRACTIE
                        IS. SPATIES EN NIET-SIGNIFICANTE NULLEN WORDEN NIET GEPONST.
                        INDIEN X≥ ₁₀N IN WAARDE DAN WORDT EEN FLOATING POINT GETAL GEPONST MET N+M
                        POSITIES VOOR DE MANTISSE.
                        DEZE PROCEDURE GEBRUIKT PONSNR(N,M);

                REAL XHULP;
                INTEGER INTPART,DECPART,MANT,EXP,R,S;
                XHULP:=ABS(X);
                IF X ≥ 0 THEN PUNCH(PLUS) ELSE PUNCH(MIN);
                XHULP:=ENTIER(XHULP*10↑M+0.5)*10↑(-M);
                IF XHULP = 0 THEN BEGIN PUNCH(PONSTABEL[0]); GOTO ENDZUINIG; END:
                IF XHULP ≥ 10↑N THEN GOTO FLOATING;
                INTPART:=ENTIER(XHULP); DECPART:=(XHULP-INTPART)*10↑M;

                IF INTPART > 0 THEN BEGIN       R:=0;
TESTINT:
                                                R:=R+1; IF INTPART ≥ 10↑R THEN GOTO TESTINT;
                                                PONSNR(R,INTPART);
                                     END;
                IF DECPART > 0 THEN BEGIN PUNCH(PUNT); PONSNR(M,DECPART); END;
                GOTO ENDZUINIG;

FLOATING:       PUNCH(PUNT);
                EXP:=N;
FINDEXP:        EXP:=EXP+1; IF XHULP ≥ 10↑EXP THEN GOTO FINDEXP;
                XHULP:=XHULP*10↑(N+M-EXP);
                IF XHULP ≥ 10↑(N+M) THEN BEGIN  EXP:=EXP+1; XHULP:=XHULP/10; END;
```

```
                MANT:=ENTIER(XHULP+0.5);
                PONSNR(N+M,MANT); PUNCH(TIEN); PUNCH(PLUS);
                R:=0;
TESTEXP:        R:=R+1; IF EXP ≥ 10↑R THEN GOTO TESTEXP;
                PONSNR(R,EXP);
ENDZUINIG:
        END PONSZUINIG;

        PROCEDURE PONSNR(N,M); VALUE N,M; INTEGER N,M;
        BEGIN INTEGER S;

NEXT:           S:=M÷10↑(N-1);
                PUNCH(PONSTABEL[S]);
                M:=M-S*10↑(N-1);
                N:=N-1;
                IF N > 0 THEN GOTO NEXT;
        END PONSNR;
        INTEGER ARRAY PONSTABEL[0:9];
        INTEGER       PLUS,MIN,PUNT,TIEN,LOWER;

        BEGIN   COMMENT VULBLOK PONSGEGEVENS;
                INTEGER Q;

                PLUS:=112; MIN:=64; PUNT:=107;
                TIEN:=59; LOWER:=122;

                PONSTABEL[0]:=32;
                FOR Q:=1,2,4,7,8 DO PONSTABEL[Q]:=Q;
                FOR Q:=3,5,6,9 DO PONSTABEL[Q]:=Q+16;
        END PONSVULBLOK;

                BEGIN COMMENT IN DIT BLOK VINDT DE OVERDRACHT VAN DE ADMINISTRATIE E.D.
                              PLAATS. VOORTS WORDT DE HEADING GEPRINT;
                      INTEGER ARRAY ADMIN[1:25];
                      REAL ARRAY YK[1:25];
                      INTEGER H;

                      TYD[1]:=TIME;
                      INARRAY(DRUM,0,ADMIN);
                      INARRAY(DRUM,40,YK);
                      INARRAY(DRUM,25,TWBR);
                      INARRAY(DRUM,90,AFW);
                      CARRIAGE(6);
                      PRINTTEXT(¢PRAG-DATA-COMPUTATION$);  SPACE(90); PRINTTEXT(¢VERSIE:$);
                      ABSFIXT(10,0,VERSIE); CARRIAGE(6);
                      PRINTTEXT(¢K.N.M.I.    DE BILT    NETHERLANDS$);  CARRIAGE(6);
                      PRINTTEXT(¢        CAMPAIGN         STATION/    INSTRUMENT      WATER        INSTRUMENT    INSTRUMENT$);
                      NLCR;
                      PRINTTEXT(¢                       .   PERIOD      DEPTH(M)      DEPTH(M)          TYPE       NUMBER$);
                      NLCR;  NLCR;
                      HOLD(ADMIN);

                      IF ADMIN[9]>2 THEN BEGIN       PRINTTEXT(¢ FOUTE P-BAND:$); ABSFIXT(4,0,ADMIN[9]); NLCR;
                                          TELETEXT(¢ DIT IS PRAG-COMPUTATION. IS NIET GEVRAAGD. ZIE HIERBOVEN$);
                                          PONS:=FALSE;
                                          GOTO STOP;
                                   END;

                      FOR J:=1,2,5,6,7,8 DO ABSFIXT(12,0,ADMIN[J]);
                      CARRIAGE(4);
```

```
                                SPACE(13); PRINTTEXT(‹DEGR  MIN      DEGR  MIN›);  NLCR;
                                PRINTTEXT(‹POSITION:›);
                                H:=ABS(ADMIN[3]);  ABSFIXT(6,0,H÷100);  ABSFIXT(4,0,H-H÷100*100);
                                IF ADMIN[3] > 0 THEN PRINTTEXT(‹N›) ELSE PRINTTEXT(‹S›);
                                H:=ABS(ADMIN[4]);  ABSFIXT(6,0,H÷100);  ABSFIXT(4,0,H-H÷100*100);
                                IF ADMIN[4] > 0 THEN PRINTTEXT(‹E›)  ELSE PRINTTEXT(‹W›);

                                CARRIAGE(4);  HOLD(YK);
                                PRINTTEXT(‹TIME OF FIRST MEASUREMENT  GMT:›); ABSFIXT(10,0,ADMIN[12]);  ABSFIXT(6,0,YK[21]);
                                NLCR;
                                PRINTTEXT(‹TIME OF  LAST MEASUREMENT  GMT:›); ABSFIXT(10,0,ADMIN[13]);  ABSFIXT(6,0,YK[22]);
                                NLCR; NLCR;
                                PRINTTEXT(‹TIME OF FIRST MEASUREMENT UNDER WATER GMT:›); ABSFIXT(10,0,ADMIN[22]): ABSFIXT(6,0,YK[23]);
                                CARRIAGE(4);

                                PRINTTEXT(‹TIME INTERVAL IN MINUTES:›);  ABSFIXT(2,0,ADMIN[11]);  CARRIAGE(4):
                                IF ADMIN[9]=0 THEN PRINTTEXT(‹NO ›);  PRINTTEXT(‹OUTPUT-TAPE IS REQUESTED›):
                                CARRIAGE(4);

                                PRINTTEXT(‹CALIBRATIONS USED›);  NLCR;  NLCR;
                                PRINTTEXT(‹GAP       TEMPERATURE
                                                                     COMPASS                      SPEED›)
                                AANDERAA:=ADMIN[7]≠11∧ADMIN[7]≠12;
                                IF AANDERAA THEN PRINTTEXT(‹
                                                                     SALINITY                     DEPTH›);  NLCR:  NLCR;
                                ABSFIXT(2,0,ADMIN[16]);
                                FOR J:=1 STEP 1 UNTIL 6 DO FIXT(3,5,YK[J]);
                                IF AANDERAA THEN BEGIN FOR J:=11 STEP 1 UNTIL 14 DO FIXT(2,7,YK[J]); END;
OVERDRACHT:
                                METNO:=ADMIN[8];  PONS:= ADMIN[9]≠0;
                                V :=ADMIN[11]; JUMP:=1023+ADMIN[16];
                                S:=ADMIN[21];  DAG:=ADMIN[22];
                                HOLD(TWBR);

                                TW:=TWBR[13]: TOTAL:=TWBR[11];  TWFIRST:=TWBR[12];  TWG:=TWBR[1];  TEND:=1;
                                COMPL:=  TWG=TOTAL;  TGW:=0;
                                A0:=YK[1];  A1:=YK[2];   KF:=YK[4];   KV:=YK[3];   APRO:=YK[5];   CFS:=YK[6];
                                IF AANDERAA THEN BEGIN B0:=YK[11];  B1:=YK[12];  C0:=YK[13];  C1:=YK[14];  END;
                                TIJD:=YK[23];
                                HOLD(AFW);
                                NLCR; NLCR; PRINTTEXT(‹COMPASS CORRECTIONS:›); NLCR;
                                FOR J:=0 STEP 1 UNTIL 8 DO FIXT(3,0,AFW[J]);

                                I:=0;  COMMENT I IS DE TELLER VAN HET ERROR ARRAY BIJ DE CORRECTIE;
                                TYD[2]:=TIME; ONE:=TRUE;
                                IF PONS THEN
                                BEGIN   RUNOUT; RUNOUT; PUTEXT(‹'CAMP,STAT'›); FIXP(4,0,ADMIN[1]); FIXP(4,0,ADMIN[2]); PUNLCR; END;
                        END OVERDRACHTBLOK;

FILLERROR:              BEGIN COMMENT IN DIT BLOK WORDT DE FOUTENTABEL OPGEBOUWD DOOR ONDERZOEK VAN
                                 ONDERZOEKT DE SIGNALERING "ACHTERIN" HET AT-ARRAY ZOALS DAT NA AFLOOP VAN KRAAK OP DE TROMMEL
                                STAAT EN BOUWT DE ERROR-TABEL OP OP DE TROMMEL TE BEGINNEN OP ADRES 120000.
                                ALS ELEMENT 0 EN ELEMENT TOTERR WORDT INGEVULD 0 EN TOTEL+1:
                        INTEGER I,J,TABLE,TEL,ADRES,HULP,ERADR,EIND;
                        INTEGER ARRAY ERROR[1:10],WIG,WAG[1:200];

                        NEWPAGE; PRINTTEXT(‹FOUTENTABEL›);
                        IF ONE THEN PRINTTEXT(‹ SNELHEID›)  ELSE PRINTTEXT(‹ RICHTING›);  NLCR:
                        TEL:=TWFIRST; J:=1; ERROR[1]:=0; TABLE:=0; EIND:=TWG; ADRES:=200+5*TWFIRST-5; ERADR:=120000;
                        INARRAY(DRUM,ADRES,WIG); ADRES:=ADRES+200;
                TESTWIG:        INARRAY(DRUM,ADRES,WAG); ADRES:=ADRES+200; HOLD(WIG);
                        FOR I:= 5 STEP 5 UNTIL 200 DO
```

```
                        BEGIN   HULP:=WIG[I];
                                IF (ONE ∧ ( HULP=1 ∨ HULP > 99 )) ∨ ( ¬ONE ∧ ( HULP≠0 ∧ HULP≠100 )) THEN
                                BEGIN   J:=J+1: ERROR[J]:=TEL; ABSFIXT(6,0,TEL); IF J=10 THEN
                                        BEGIN   OUTARRAY(DRUM,ERADR,ERROR); ERADR:=ERADR+10; TABLE:=TABLE+10: NLCR; J:=0; HOLD(ERROR);
                                        END;
                                END FOUTWIG;
                                TEL:=TEL+1; IF TEL>EIND THEN GOTO ENDFILL;
                        END LOOPWIG;
                TESTWAG:        INARRAY(DRUM,ADRES,WIG); ADRES:=ADRES+200; HOLD(WAG);
                        FOR I:= 5 STEP 5 UNTIL 200 DO
                        BEGIN   HULP:=WAG[I];
                                IF ( ONE ∧ ( HULP=1 ∨ HULP > 99 )) ∨ ( ¬ONE ∧ ( HULP ≠0  ∧ HULP ≠ 100 )) THEN
                                BEGIN   J:=J+1; ERROR[J]:=TEL; ABSFIXT(6,0,TEL); IF J=10 THEN
                                        BEGIN   OUTARRAY(DRUM,ERADR,ERROR); ERADR:=ERADR+10; TABLE:=TABLE+10; NLCR; J:=0; HOLD(ERROR);
                                        END;
                                END FOUTWAG;
                                TEL:=TEL+1; IF TEL>EIND THEN GOTO ENDFILL;
                        END LOOPWAG;
                        GOTO TESTWIG;
                ENDFILL:        TABLE:=TABLE+J;  COMMENT ALS ER GEEN FOUTEN GEVONDEN ZIJN IS TABLE=J=1;
                                IF TABLE=1 THEN BEGIN   PRINTTEXT(‹GEEN FOUTEN GEVONDEN›); NLCR;
                                                        IF ONE THEN     BEGIN ONE:= FALSE;
                                                                              GOTO FILLERROR;
                                                                        END
                                                                        ELSE GOTO ACORRECT;
                                                END;
                                FOR J:= J+1 STEP 1 UNTIL 10 DO
                        ERROR[J]:=EIND+1; OUTARRAY(DRUM,ERADR,ERROR); TOTERR:=TABLE; HOLD(ERROR); TYD[3]:=TIME;
                END FILLERROR;

                CORRECTIE:      BEGIN INTEGER ARRAY ERROR[0:TOTERR];
                                        INTEGER HULP,VAR,HELP,ARHULP,M,AAND,HERR;
                                        REAL ARRAY COEF[1:6]:

                                        BOOLEAN PROCEDURE GAP(IHULP); INTEGER IHULP;
                                        GAP:=(IHULP=0 ∨ IHULP=1  ∨  IHULP=1023):

                PROCEDURE SOLVE(A,X,G,N,M, SKIP); VALUE N,M; REAL ARRAY A,X,G; INTEGER N,M; LABEL SKIP;
                BEGIN   COMMENT LOST HET STELSEL VERGELIJKINGEN A.X=G OP DOOR ELIMINATIE;
                        INTEGER I,J; REAL HULP;
                        IF M=1 THEN X[1]:=G[1]/A[1,1] ELSE
                        BEGIN   J:=0;
                        SOLVETEST: IF ABS(A[M,M])<⏨-6 ∨ ABS(A[M,M]*A[M-1,M-1]-A[M-1,M]*A[M,M-1])<⏨-6 THEN
                                BEGIN   J:=J+1;
                                        IF J=M THEN BEGIN PRINTTEXT(‹STELSEL NIET OPLOSBAAR›);
                                                          FOR I:=1 STEP 1 UNTIL N DO X[I]:=0; NLCR; GOTO SKIP:
                                                    END;
                                        HULP:=G[J]; G[J]:=G[M]; G[M]:=HULP;
                                        FOR I:= 1 STEP 1 UNTIL M DO
                                        BEGIN   HULP:=A[I,J]; A[I,J]:=A[I,M]; A[I,M]:=HULP;
                                        END;
                                        GOTO SOLVETEST;
                                END;
                                FOR I:= 1 STEP 1 UNTIL M-1 DO IF ABS(A[M,I])>⏨-6 THEN
                                BEGIN   G[I]:=G[M]-A[M,M]*G[I]/A[M,I];
                                        FOR J:= 1 STEP 1 UNTIL M-1 DO
                                        A[J,I]:=A[J,M]-A[J,I]*A[M,M]/A[M,I];
                                END;
                                SOLVE(A,X,G,N,M-1, SKIP);
                                X[M]:=(G[M]-SUM(I,1,M-1,A[I,M]*X[I]))/A[M,M];
```

```
                          SPACE(13); PRINTTEXT(‹DEGR  MIN       DEGR  MIN›);  NLCR;
                          PRINTTEXT(‹POSITION:›);
                          H:=ABS(ADMIN[3]);  ABSFIXT(6,0,H÷100);  ABSFIXT(4,0,H-H÷100*100);
                          IF ADMIN[3] > 0 THEN PRINTTEXT(‹N›) ELSE PRINTTEXT(‹S›);
                          H:=ABS(ADMIN[4]);  ABSFIXT(6,0,H÷100);  ABSFIXT(4,0,H-H÷100*100);
                          IF ADMIN[4] > 0 THEN PRINTTEXT(‹E›)  ELSE PRINTTEXT(‹W›);

                          CARRIAGE(4);  HOLD(YK);
                          PRINTTEXT(‹TIME OF FIRST MEASUREMENT  GMT:›); ABSFIXT(10,0,ADMIN[12]);  ABSFIXT(6,0,YK[21]);
                          NLCR;
                          PRINTTEXT(‹TIME OF  LAST MEASUREMENT  GMT:›); ABSFIXT(10,0,ADMIN[13]);  ABSFIXT(6,0,YK[22]);
                          NLCR; NLCR;
                          PRINTTEXT(‹TIME OF FIRST MEASUREMENT UNDER WATER GMT:›); ABSFIXT(10,0,ADMIN[22]): ABSFIXT(6,0,YK[23]);
                          CARRIAGE(4);

                          PRINTTEXT(‹TIME INTERVAL IN MINUTES:›):  ABSFIXT(2,0,ADMIN[11]);  CARRIAGE(4):
                          IF ADMIN[9]=0 THEN PRINTTEXT(‹NO ›);  PRINTTEXT(‹OUTPUT-TAPE IS REQUESTED›):
                          CARRIAGE(4);

                          PRINTTEXT(‹CALIBRATIONS USED›);  NLCR;  NLCR;
                          PRINTTEXT(‹GAP       TEMPERATURE
                          AANDERAA:=ADMIN[7]≠11∧ADMIN[7]≠12;               COMPASS                    SPEED›);
                          IF AANDERAA THEN PRINTTEXT(‹
                          ABSFIXT(2,0,ADMIN[16]);                         SALINITY
                                                                                                       DEPTH›);  NLCR:  NLCR;
                          FOR J:=1 STEP 1 UNTIL 6 DO FIXT(3,5,YK[J]);
                          IF AANDERAA THEN BEGIN FOR J:=11 STEP 1 UNTIL 14 DO FIXT(2,7,YK[J]); END;
OVERDRACHT:
                          METNO:=ADMIN[8];  PONS:= ADMIN[9]≠0;
                          V :=ADMIN[11]; JUMP:=1023+ADMIN[16];
                          S:=ADMIN[21];  DAG:=ADMIN[22];
                          HOLD(TWBR);

                          TW:=TWBR[13]; TOTAL:=TWBR[11];  TWFIRST:=TWBR[12];  TWG:=TWBR[1];  TEND:=1;
                          COMPL:=  TWG=TOTAL;  TGW:=0;
                          A0:=YK[1];  A1:=YK[2];   KF:=YK[4];   KV:=YK[3];   APRO:=YK[5];   CFS:=YK[6];
                          IF AANDERAA THEN BEGIN B0:=YK[11];  B1:=YK[12];  C0:=YK[13];  C1:=YK[14];  END;
                          TIJD:=YK[23];
                          HOLD(AFW);
                          NLCR; NLCR; PRINTTEXT(‹COMPASS CORRECTIONS:›); NLCR;
                          FOR J:=0 STEP 1 UNTIL 8 DO FIXT(3,0,AFW[J]);

                          I:=0;  COMMENT I IS DE TELLER VAN HET ERROR ARRAY BIJ DE CORRECTIE;
                          TYD[2]:=TIME; ONE:=TRUE;
                          IF PONS THEN
                          BEGIN   RUNOUT; RUNOUT; PUTEXT(‹'CAMP,STAT'›); FIXP(4,0,ADMIN[1]); FIXP(4,0,ADMIN[2]); PUNLCR; END;
                   END OVERDRACHTBLOK;
FILLERROR:         BEGIN COMMENT IN DIT BLOK WORDT DE FOUTENTABEL OPGEBCUWD DOOR ONDERZOEK VAN
                          ONDERZOEKT DE SIGNALERING "ACHTERIN" HET AT-ARRAY ZOALS DAT NA AFLOOP VAN KRAAK OP DE TROMMEL
                          STAAT EN BOUWT DE ERROR-TABEL OP OP DE TROMMEL TE BEGINNEN OP ADRES 120000.
                          ALS ELEMENT 0 EN ELEMENT TOTERR WORDT INGEVULD 0 EN TOTEL+1;
                   INTEGER I,J,TABLE,TEL,ADRES,HULP,ERADR,EIND;
                   INTEGER ARRAY ERROR[1:10],WIG,WAG[1:200];

                   NEWPAGE; PRINTTEXT(‹FOUTENTABEL›);
                   IF ONE THEN PRINTTEXT(‹ SNELHEID›)  ELSE PRINTTEXT(‹ RICHTING›);  NLCR:
                   TEL:=TWFIRST; J:=1; ERROR[1]:=0; TABLE:=0; EIND:=TWG; ADRES:=200+5*TWFIRST-5; ERADR:=120000;
                   INARRAY(DRUM,ADRES,WIG); ADRES:=ADRES+200;
           TESTWIG:       INARRAY(DRUM,ADRES,WAG); ADRES:=ADRES+200; HOLD(WIG);
                   FOR I:= 5 STEP 5 UNTIL 200 DO
```

```
                   BEGIN   HULP:=WIG[I];
                          IF (ONE ∧ ( HULP=1 ∨ HULP > 99 )) ∨ ( ¬ONE ∧ ( HULP≠0 ∧ HULP≠100 )) THEN
                          BEGIN   J:=J+1; ERROR[J]:=TEL; ABSFIXT(6,0,TEL); IF J=10 THEN
                                 BEGIN   OUTARRAY(DRUM,ERADR,ERROR); ERADR:=ERADR+10; TABLE:=TABLE+10  NLCR; J:=0; HOLD(ERROR);
                                 END;
                          END FOUTWIG;
                          TEL:=TEL+1; IF TEL>EIND THEN GOTO ENDFILL;
                   END LOOPWIG;
           TESTWAG:       INARRAY(DRUM,ADRES,WIG); ADRES:=ADRES+200; HOLD(WAG);
                   FOR I:= 5 STEP 5 UNTIL 200 DO
                   BEGIN   HULP:=WAG[I];
                          IF ( ONE ∧ ( HULP=1 ∨ HULP > 99 )) ∨ ( ¬ONE ∧ ( HULP ≠0  ∧ HULP ≠ 100 )) THEN
                          BEGIN   J:=J+1; ERROR[J]:=TEL; ABSFIXT(6,0,TEL); IF J=10 THEN
                                 BEGIN   OUTARRAY(DRUM,ERADR,ERROR); ERADR:=ERADR+10; TABLE:=TABLE+10; NLCR; J:=0; HOLD(ERROR);
                                 END;
                          END FOUTWAG;
                          TEL:=TEL+1; IF TEL>EIND THEN GOTO ENDFILL;
                   END LOOPWAG;
                   GOTO TESTWIG;
           ENDFILL:       TABLE:=TABLE+J;  COMMENT ALS ER GEEN FOUTEN GEVONDEN ZIJN IS TABLE=J=1;
                          IF TABLE=1 THEN BEGIN   PRINTTEXT(‹GEEN FOUTEN GEVONDEN›); NLCR;
                                                 IF ONE THEN     BEGIN ONE:= FALSE;
                                                                        GOTO FILLERROR;
                                                                 END
                                                                 ELSE GOTO ACORRECT;
                                         END;
                          FOR J:= J+1 STEP 1 UNTIL 10 DO
                   ERROR[J]:=EIND+1; OUTARRAY(DRUM,ERADR,ERROR); TOTERR:=TABLE; HOLD(ERROR); TYD[3]:=TIME;
           END FILLERROR;

           CORRECTIE:     BEGIN INTEGER ARRAY ERROR[0:TOTERR];
                                 INTEGER HULP,VAR,HELP,ARHULP,M,AAND,HERR;
                                 REAL ARRAY COEF[1:6];

                                 BOOLEAN PROCEDURE GAP(IHULP): INTEGER IHULP;
                                 GAP:=(IHULP=0 ∨ IHULP=1  ∨  IHULP=1023);
           PROCEDURE SOLVE(A,X,G,N,M, SKIP); VALUE N,M; REAL ARRAY A,X,G; INTEGER N,M; LABEL SKIP;
           BEGIN   COMMENT LOST HET STELSEL VERGELIJKINGEN A.X=G OP DOOR ELIMINATIE;
                   INTEGER I,J; REAL HULP;
                   IF M=1 THEN X[1]:=G[1]/A[1,1] ELSE
                   BEGIN   J:=0;
                   SOLVETEST: IF ABS(A[M,M])<⏨-6 ∨ ABS(A[M,M]*A[M-1,M-1]-A[M-1,M]*A[M,M-1])<⏨-6 THEN
                          BEGIN   J:=J+1;
                                 IF J=M THEN BEGIN PRINTTEXT(‹STELSEL NIET OPLOSBAAR›);
                                                   FOR I:=1 STEP 1 UNTIL N DO X[I]:=0; NLCR; GOTO SKIP:
                                            END;
                                 HULP:=G[J]; G[J]:=G[M]; G[M]:=HULP;
                                 FOR I:= 1 STEP 1 UNTIL M DO
                                 BEGIN   HULP:=A[I,J]; A[I,J]:=A[I,M]; A[I,M]:=HULP;
                                 END;
                                 GOTO SOLVETEST;
                          END;
                          FOR I:= 1 STEP 1 UNTIL M-1 DO IF ABS(A[M,I])>⏨-6 THEN
                          BEGIN   G[I]:=G[M]-A[M,M]*G[I]/A[M,I];
                                 FOR J:= 1 STEP 1 UNTIL M-1 DO
                                 A[J,I]:=A[J,M]-A[J,I]*A[M,M]/A[M,I];
                          END;
                          SOLVE(A,X,G,N,M-1, SKIP);
                          X[M]:=(G[M]-SUM(I,1,M-1,A[I,M]*X[I]))/A[M,M];
```

```
                    END;
          ENDSOLVE:
          END;

          REAL PROCEDURE POL(M,X,COEF); VALUE M,X; INTEGER M; REAL X; REAL ARRAY COEF;
          BEGIN   INTEGER I; REAL S;
                  S:=COEF[M];
                  FOR I:=M-1 STEP -1 UNTIL 1 DO S:=S*X+COEF[I];
                  POL:=S;
          END POL;

          PROCEDURE KKPOL(X,Y,N,M,COEF,SKIP); VALUE N,M; INTEGER N,M; REAL ARRAY X,Y,COEF; LABEL SKIP;
          BEGIN   COMMENT KKPOL BEREKENT DE KLEINSTE KWADRATEN AANPASSING DOOR N PUNTEN X,Y
                          MET EEN GRAAD VAN TEN HOOGSTE M-1 EN LEVERT HET RESULTAAT AF IN COEF;
                  REAL ARRAY A[1:M,1:M],HCOEF,B[1:M];
                  INTEGER P,Q,R,MM;
                  REAL FOUT1,FOUT2;
                  FOUT2:=SUM(R,1,N,Y[R])/N;
                  FOUT1:=SUM(R,1,N,(Y[R]-FOUT2)↑2);
                  COEF[1]:=FOUT2;
                  FOR P:=2 STEP 1 UNTIL M DO COEF[P]:=0;
                  MM:=2;
          WERK:   FOR P:= 1 STEP 1 UNTIL MM DO
                  BEGIN   HCOEF[P]:=0;
                          B[P]:=SUM(R,1,N,Y[R]*X[R]↑(P-1));
                          FOR Q:= 1 STEP 1 UNTIL MM DO
                          A[P,Q]:=SUM(R,1,N,X[R]↑(P+Q-2));
                  END;
                  SOLVE(A,HCOEF,B,MM,MM,SKIP);
                  FOUT2:=SUM(R,1,N,(Y[R]-POL(MM,X[R],HCOEF))↑2);
                  IF FOUT2< FOUT1 THEN BEGIN FOUT1:=FOUT2;
                                             FOR R:= 1 STEP 1 UNTIL MM DO COEF[R]:=HCOEF[R];
                                             MM:=MM+1;
                                             IF MM≤M THEN GOTO WERK;
                                       END;
          END KKPOL;


                INARRAY(DRUM,120000,ERROR); HOLD(ERROR); HERR:=TOTERR; AAND:= IF AANDERAA THEN 2 ELSE 0:
TESTER:         IF ERROR[TOTERR]-TWG≥0 THEN
                BEGIN   IF ERROR[TOTERR]=TWG THEN TWG:=TWG-1;
                        TOTERR:=TOTERR-1; IF TOTERR=0 THEN GOTO TESTONE; GOTO TESTER
                END;
                IF I=TOTERR THEN GOTO SETM; NEWPAGE;
                IF ONE THEN PRINTTEXT(‘SPEED ’)  ELSE  PRINTTEXT(‘DIRECTION ’);
                PRINTTEXT(‘INTERPOLATIONS AT:’); NLCR;
                PRINTTEXT(‘   NR    OLD    NEW   SORT’); NLCR;
TESTI:          IF ERROR[I] ≤ TWFIRST THEN BEGIN I:=I+1; GOTO TESTI; END;
                COMMENT HET EERSTE SNELHEIDS/RICHTINGS-GETAL WORDT ALS JUIST AANGENOMEN;
SETM:           M:=ERROR[I]-5;
                IF M<TWFIRST THEN BEGIN M:=TWFIRST; END;
                HELP:=M; ARHULP:=1; J:=0; GOTO TESTN;
TESTAR:         IF HELP+J+1=ERROR[I] THEN
                BEGIN   J:=0; HELP:=ERROR[I]; I:=I+1;
                        GOTO TESTN;
                END;

                ARHULP:=ARHULP+1; J:=J+1;
TESTN:          IF HELP+J>TWG THEN GOTO LAST;
                IF J≥5 ∧ ARHULP>9 THEN BEGIN N:=HELP+J; GOTO INTERPOL; END;
                GOTO TESTAR;
```

```

LAST:           N:=TWG;   K:=TOTERR; I:=TOTERR+1;
                HELP:=N;   J:=0; ARHULP:=1;
TESTOR:         IF J≥5 ∧ ARHULP>9 THEN
                BEGIN   M:=HELP-J; GOTO INTERPOL; END;
                J:=J+1;
                IF HELP-J≤ ERROR[K] THEN
                BEGIN   HELP:=ERROR[K];
                        K:=K-1; IF K=0 THEN
                        BEGIN   CARRIAGE(4); PRINTTEXT(‘(GEDEELTE VAN) DE REEKS IS TEKORT’); NLCR;NLCR;
                                TOTERR:=0; GOTO TESTONE;
                        END;    J:=0; GOTO TESTOR;
                END;
                ARHULP:=ARHULP+1; GOTO TESTOR;

INTERPOL:       BEGIN   INTEGER ARRAY DATA[1:5*(N-M+1)];
                        REAL ARRAY X,TIME[1:ARHULP];
                        ADR:=200+5*(M-1);
                        INARRAY(DRUM,ADR,DATA); HOLD(DATA);
                        IF M=TWFIRST THEN DATA[5]:=0; HULP:=0;
                        IF ¬ONE THEN GOTO INTRICHT;
INTSNEL:                FOR J:= 1 STEP 1 UNTIL ARHULP DO
                        BEGIN
                ATESTVLAG:
                                IF (DATA[5*(J+HULP)]=1 ∨ DATA[5*(J+HULP)]>99) ∧ J≠1 THEN
                                BEGIN   HULP:=HULP+1; GOTO ATESTVLAG END;
                                X[J]:=DATA[5*(J+HULP)-1]; TIME[J]:=J+HULP;
                        END;
                SETGAP: FOR J:=ARHULP STEP -1 UNTIL 2 DO
                        BEGIN   IF X[J]<X[J-1] THEN
                                BEGIN   FOR HULP:= J STEP 1 UNTIL ARHULP DO X[HULP]:=X[HULP]+JUMP
                                END
                        END;
                        KKPOL(TIME,X,ARHULP,6,COEF,TESTONE);
                        VAR:=SUM(J,1,ARHULP,ABS(X[J]-POL(6,TIME[J],COEF)))/ARHULP;
                        FOR J:= 1 STEP 1 UNTIL N-M+1 DO
                        BEGIN   HELP:=DATA[5*J];
                                IF HELP≠1 ∨ HELP>99 THEN
                                BEGIN   HULP:=POL(6,J,COEF);
                TESTGAP:                IF HULP>JUMP THEN BEGIN HULP:=HULP-JUMP; GOTO TESTGAP END
                                END     ELSE GOTO AFOLLOW;
                                IF HELP≠1 THEN
                                BEGIN   DATA[5*J-1]:=HULP; ABSFIXT(6,0,M+J-1); FIXT(10,0,HULP); PRINTTEXT(‘ A ’); NLCR;
                                        GOTO AFOLLOW
                                END;
                                HELP:=ABS((2-AAND)/2*1023-DATA[5*J-1]);
                                IF GAP(HELP) THEN
                                BEGIN   DATA[5*J-1]:=HULP;
                                        ABSFIXT(6,0,M+J-1);FIXT(4.0,HELP); FIXT(4,0,HULP); PRINTTEXT(‘ V ’); NLCR;
                                        GOTO AFOLLOW
                                END;
                                HELP:=DATA[5*J-1];
                                IF ABS(HELP-HULP)>2*VAR ∧ ABS(HELP-JUMP-HULP)>2*VAR THEN
                                BEGIN   DATA[5*J-1]:=HULP; ABSFIXT(6,0,M+J-1); FIXT(4,0,HELP); FIXT(4,0,HULP);
                                         PRINTTEXT(‘ V ’); NLCR;
                                END ELSE DATA[5*J]:=DATA[5*J]-100;
                AFOLLOW:
                        END;
                        GOTO OUTDATA;
INTRICHT:                       FOR J:= 1 STEP 1 UNTIL ARHULP DO
```

```
                          BEGIN
                 BTESTVLAG:
                                  HELP:=DATA[5*(J+HULP)];
                                  IF HELP≠0 ∧ J≠1 ∧ HELP≠100 THEN
                                  BEGIN   HULP:=HULP+1; GOTO BTESTVLAG END;
                                  X[J]:=DATA[5*(J+HULP)-2];
                                  TIME[J]:=J+HULP;
                          END:
                 GAPSET:  FOR J:= 1 STEP 1 UNTIL ARHULP-1 DO
                          BEGIN   IF X[J+1]-X[J]>520 THEN X[J+1]:=X[J+1]-1037;
                                  IF X[J]-X[J+1]>520 THEN X[J+1]:=X[J+1]+1037;
                          END:
                          KKPOL(TIME,X,ARHULP,6,COEF,TESTONE);
                          VAR:=SUM(J,1,ARHULP,ABS(X[J]-POL(6,TIME[J],COEF)))/ARHULP;
                          FOR J:= 1 STEP 1 UNTIL N-M+1 DO
                          BEGIN   HELP:=DATA[5*J];
                                  IF HELP≠0 ∧ HELP≠100 THEN
                                  BEGIN . HULP:=POL(6,J,COEF);
                 GAPTEST:                 IF HULP<0 THEN HULP:=HULP+1037;
                                          IF HULP>1037 THEN HULP:=HULP-1037;
                                  END ELSE GOTO BFOLLOW;
                                  IF HELP=1 THEN BEGIN DATA[5*J-2]:=HULP; ABSFIXT(6,0,M+J-1); FIXT(10,3,HULP):
                                                            PRINTTEXT(‹ A ›); NLCR; GOTO BFOLLOW END:
                                  HELP:=DATA[5*J-2];
                                  IF GAP(HELP) THEN
                                  BEGIN   DATA[5*J-2]:=HULP;
                                          ABSFIXT(6,0,M+J-1);FIXT(4,0,HELP); FIXT(4,0,HULP);
                                          PRINTTEXT(‹ D ›); NLCR; GOTO BFOLLOW
                                  END;
                          IF ABS(HELP-HULP)>2*VAR ∧ ABS(ABS(HELP-HULP)-1037)>2*VAR THEN
                                  BEGIN   ABSFIXT(6,0,M+J-1);FIXT(4,0,HELP); FIXT(4,0,HULP);
                                          DATA[5*J-2]:=HULP; PRINTTEXT(‹ D ›); NLCR;
                                  END ELSE DATA[5*J]:=DATA[5*J]-10;
                 BFOLLOW:
                          END;
        OUTDATA:          OUTARRAY(DRUM,ADR,DATA); HOLD(DATA);
                 END      DATA BLOK;
        TESTONE:          IF I≤TOTERR∧TOTERR ≠ 0 THEN GOTO SETM;
                          IF ONE THEN BEGIN ONE:=FALSE;  I:=0;  GOTO FILLERROR; END:

        END      CORRECTIE BLOK;
ACORRECT:
        N:=60/V;
        TYD[4]:=TIME;
        UURADR:=140800;TOTUUR:=0;
        RAD:=ARCTAN(1)/45;
        BEGIN   REAL GEMTEMP,GEMOOST,GEMNOORD,TEMP,OOST,NOORD;

        PROCEDURE TEMPERATURE(G,TEMP); INTEGER G; REAL TEMP;
        BEGIN   TEMP:=A0+A1*G
        END TEMPERATURE;

        PROCEDURE SALINITY(G,SAL); INTEGER G; REAL SAL;
        BEGIN   SAL:=B0+B1*G
        END SALINITY;

        PROCEDURE DIEPTE(G,DIEP); INTEGER G; REAL DIEP;
        BEGIN   DIEP:=(C0+C1*G)*10
        END DIEPTE;

```

```
        PROCEDURE DIRECTION(G,RICHT); INTEGER G; REAL RICHT;
        BEGIN   INTEGER I; REAL D,DEV;
                RICHT:=KF*G+5;
                IF RICHT<0 THEN RICHT:=RICHT+360;
                IF RICHT≥360 THEN RICHT:=RICHT-360;
                D:=RICHT/45; I:=ENTIER(D); IF I>7 ∨ I<0 THEN
                BEGIN CARRIAGE(4); PRINTTEXT(‹FOUT RICHTINGSGETAL. STOP NA:›);ABSFIXT(8,0,TIME); GOTO STOP; END;
                DEV:=AFW[I]+(D-I)*(AFW[I+1]-AFW[I]);
                RICHT:=RICHT+DEV+KV;
                IF RICHT < 0 THEN RICHT:=RICHT+360;
                IF RICHT≥360 THEN RICHT:=RICHT-360
        END DIRECTION;

        PROCEDURE VELOCITY(G,VG,SNELH); INTEGER G,VG; REAL SNELH;
        BEGIN   INTEGER DELTA; DELTA:=G-VG;
                IF DELTA<0 THEN DELTA:=DELTA+JUMP;
                SNELH:=APRO+CFS*(DELTA/V);
        END VELOCITY;

        PROCEDURE GEMRICHT(VR,RICHT,GEMR); REAL VR,RICHT,GEMR;
        BEGIN   IF ABS(VR-RICHT)>180 THEN
                BEGIN   IF VR>RICHT THEN RICHT:=RICHT+360 ELSE VR:=VR+360 END;
                GEMR:=(VR+RICHT)/2;
                IF GEMR≥360 THEN GEMR:=GEMR-360;
                IF RICHT≥360 THEN RICHT:=RICHT-360;
                VR:=RICHT
        END GEMRICHT;

        PROCEDURE HERSTEL(A,B); REAL A,B;
        BEGIN   IF A≥360 THEN A:=A-360; IF A<0 THEN A:=A+360;
                IF B≥360 THEN B:=B-360; IF B<0 THEN B:=B+360;
        END HERSTEL;

        PROCEDURE TRANSF(OOST,NOORD,SNELH,RICHT);
        REAL OOST,NOORD,SNELH,RICHT;
        BEGIN   IF NOORD=0 THEN
                BEGIN   IF OOST=0 THEN RICHT:=0 ELSE
                        IF OOST>0 THEN RICHT:=90 ELSE
                        IF OOST<0 THEN RICHT:=270
                END ELSE
                IF (OOST≤0 ∧ NOORD<0) ∨ ( OOST≥0 ∧ NOORD<0) THEN
                RICHT:=ARCTAN(OOST/NOORD)/ RAD+180 ELSE
                IF OOST≤0 ∧ NOORD>0 THEN
                RICHT:=ARCTAN(OOST/NOORD)/RAD+360 ELSE
                RICHT:=ARCTAN(OOST/NOORD)/RAD;
                SNELH:=SQRT(OOST*OOST+NOORD*NOORD)
        END TRANSF;

        PROCEDURE PLOT(D,V); REAL D,V;
        BEGIN   INTEGER SP1,SP2;
                SP1:=D/10; SP2:=V/2;
                IF SP2>SP1 THEN
                BEGIN   SPACE(SP1); PRINTTEXT(‹+›);
                        IF SP2≤58 THEN BEGIN SPACE(SP2-SP1-1); PRINTTEXT(‹.›) END ELSE
                        BEGIN SPACE(58-SP1-1); PRINTTEXT(‹V›) END
                END ELSE
                IF SP2<SP1 THEN BEGIN SPACE(SP2); PRINTTEXT(‹.›); SPACE(SP1-SP2-1); PRINTTEXT(‹+›) END
                ELSE BEGIN SPACE(SP1); PRINTTEXT(‹S›) END
        END PLOT;

```

```
          INTEGER PROCEDURE IMOD(INT,GETAL); VALUE INT,GETAL; INTEGER INT,GETAL;
          BEGIN   IMOD:=INT-INT÷GETAL*GETAL END;

          REAL PROCEDURE RMOD(REAL,GETAL); VALUE REAL,GETAL; REAL REAL,GETAL;
          BEGIN   RMOD:=REAL-ENTIER(REAL/GETAL)*GETAL END;

          PROCEDURE TIMING(TIJD); REAL TIJD;
          BEGIN   IF RMOD(TIJD,100)≥60 THEN TIJD:=TIJD+40;
                  IF TIJD≥2400 THEN BEGIN TIJD:=TIJD-2400; DAG:=DAG+1 END;
          END TIMING;

          PROCEDURE DAGTEL;
          BEGIN   INTEGER HDAG,MND;
                  HDAG:=IMOD(DAG,100);
                  IF HDAG≤28 THEN GOTO DAGKLAAR;
                  MND:=IMOD((DAG-HDAG)÷100,100); IF MND=2 THEN
                  BEGIN   IF IMOD(DAG÷10000,4)=0 THEN
                          BEGIN   IF HDAG=30 THEN DAG:=DAG+71; GOTO DAGKLAAR END
                          ELSE BEGIN DAG:=DAG+72; GOTO DAGKLAAR END;
                  END MND=2;
                  IF MND=1 ∨ MND=3 ∨ MND=5 ∨ MND=7 ∨ MND=8 ∨ MND=10 ∨ MND=12 THEN
                  BEGIN   IF HDAG=32 THEN
                          BEGIN   DAG:=DAG+69; IF MND=12 THEN DAG:=DAG+8800; END;
                          GOTO DAGKLAAR;
                  END;
                  IF HDAG=31 THEN DAG:=DAG+70;
          DAGKLAAR:
          END DAGTEL;

          PROCEDURE KOP;
          BEGIN   PRINTTEXT(‹ NR   REF  TEMPERATURE    DIRECTION    VELOCITY     EAST  NORTH›); SPACE(16);
                  PRINTTEXT(‹+0        +90       +180       +270      +360
                  PRINTTEXT(‹           CH1     CH2               CH3/5 DEGR    CH4/6 CM/S   COMP   COMP›); SPACE(16);
                  PRINTTEXT(‹.0            .20         .40          .60         .80         .100      CM/S›); NLCR
          END KOP;

          PROCEDURE UURKOP;
          BEGIN   PRINTTEXT(‹   DATE     HR MTEMP  MSAL MDEPTH EAST    NORTH  DIR  VELOC›); SPACE(15);
                  PRINTTEXT(‹+0        +90       +180       +270      +360
                  SPACE(33);                                                                DEGR›); NLCR;
                  PRINTTEXT(‹COMP     COMP DEGR    CM/S›); SPACE(15);
                  PRINTTEXT(‹.0            .20         .40          .60         .80         .100      CM/S›); NLCR;
          END     UURKOP;

          BOOLEAN BLOOP,SDBOOL;
          INTEGER LOOP,SDLOOP,NRNEXT,GSAL,GDIEP;
          INTEGER ARRAY WIG,WAG[1:200],ASD,BSD[1:80];

          PROCEDURE INITNEXT(NUMMER); VALUE NUMMER; INTEGER NUMMER;
          BEGIN   COMMENT MAAKT ALLES GEREED VOOR DE EERSTE AANROEP VAN NEXT, DIE DAN
                  ALS EERSTE DE MET'NUMMER' GEGEVEN METING LEVERT;
                  NRNEXT:=NUMMER-1;
                  ADR:=200+5*NUMMER-5; IF ADR > 119999 THEN BREEKAF(‹ADRESFOUT IN INITNEXT›);
                  INARRAY(DRUM,ADR,WIG); ADR:=ADR+200;
                  INARRAY(DRUM,ADR,WAG); ADR:=ADR+200;
                  LOOP:=0;
                  BLOOP:=FALSE;
                  HOLD(WIG);
          END INITNEXT;
```

```
          PROCEDURE NEXT;
          BEGIN   COMMENT IN A WORDT DE VOLGENDE METING AFGEGEVEN ZOALS DIE VOLGT UIT DE
                  BUFFERS WIG EN WAG DIE VAN AF DE TROMMEL GEVULD WORDEN;
                  NRNEXT:=NRNEXT+1;
                  IF BLOOP THEN GOTO BUFFERWAG;
 BUFFERWIG:       FOR J:=1 STEP 1 UNTIL 5 DO A[J]:=WIG[LOOP+J];
                  LOOP:=LOOP+5;
                  IF LOOP < 200 THEN GOTO ENDNEXT;
                  IF ADR > 119999 THEN BREEKAF(‹ADRESFOUT IN NEXT›);
                  INARRAY(DRUM,ADR,WIG); ADR:=ADR+200;
                  LOOP:=0; BLOOP:=TRUE;
                  HOLD(WAG);
                  GOTO ENDNEXT;
 BUFFERWAG:       FOR J:=1 STEP 1 UNTIL 5 DO A[J]:=WAG[LOOP+J];
                  LOOP:=LOOP+5;
                  IF LOOP < 200 THEN GOTO ENDNEXT;
                  IF ADR > 119999 THEN BREEKAF(‹ADRESFOUT IN NEXT›);
                  INARRAY(DRUM,ADR,WAG); ADR:=ADR+200;
                  LOOP:=0; BLOOP:=FALSE;
                  HOLD(WIG):
 ENDNEXT: END NEXT;

          PROCEDURE SDNEXT;
          BEGIN   IF SDBOOL THEN GOTO BBUF;
                  GSAL:=ASD[SDLOOP+1]; GDIEP:=ASD[SDLOOP+2];
                  SDLOOP:=SDLOOP+2;
                  IF SDLOOP=80 THEN BEGIN INARRAY(DRUM,SDADR,ASD); SDBOOL:=TRUE; SDLOOP:=0: HOLD(BSD); END:
                  GOTO SDEND;
 BBUF:            GSAL:=BSD[SDLOOP+1]; GDIEP:=ASD[SDLOOP+2];
                  SDLOOP:=SDLOOP+2;
                  IF SDLOOP=80 THEN BEGIN INARRAY(DRUM,SDADR,BSD); SDBOOL:=FALSE; SDLOOP:=0; HOLD(ASD); END:
 SDEND:   END SDNEXT;

          INITNEXT(TWFIRST);
          IF AANDERAA THEN
          BEGIN   SDADR:=80000+2*TWFIRST; INARRAY(DRUM,SDADR,ASD); SDADR:=SDADR+80;
                  INARRAY(DRUM,SDADR,BSD); SDADR:=SDADR+80; SDBOOL:=FALSE; SDLOOP:=0; HOLD(ASD);
          END;

          IF TEND=1 THEN
                  BEGIN   NEXT; VGR:=A[3]; VGS:=A[4]; VGT:=A[2];
                          TIJD:=TIJD+V; TIMING(TIJD);
                          IF (ENTIER(TIJD/100) >S ∧ S≠0) ∨ (TIJD/100<1 ∧ S=23) ∨ (S=0 ∧ TIJD>60) THEN
                          BEGIN   S:=S+1; IF S=24 THEN BEGIN  DAGTEL; S:=0 END
                          END;
                  END;
                  NEWPAGE; ABSFIXT(8,0,DAG); NLCR; KOP;
                  IF PONS THEN BEGIN RUNOUT; PUNLCR; PUMEP(LOWER); END;
                  K:=0; DTEL:=TEL:=1; VDAG:=DAG-1;
                  SAL:=DIEP:=0; COMMENT VOORBEREID DOOR FINDFIRST;
                  GEMTEMP:=GEMOOST:=GEMNOORD:=0;
                  DIRECTION(VGR,VR);
          ALLES:  NEXT;
                  IF A[1]=999999 THEN
                  BEGIN A[1]:=METNO; OUTARRAY(DRUM,200+5*(NRNEXT-1),A); TOTUUR:=TOTUUR+K:
                          OUTARRAY(DRUM,UURADR,UURW); HOLD(UURW); GOTO MALT4
                  END;
                  IF A[5]=1 THEN BEGIN A[1]:=METNO; A[2]:=VGT END;
                  ABSFIXT(3,0,DTEL); ABSFIXT(4,0,A[1]);
                  IF ABS(A[1]-METNO)<10 THEN SPACE(1) ELSE PRINTTEXT(‹R›);
                  ABSFIXT(4,0,A[2]); TEMPERATURE(A[2],TEMP); FIXT(2,1,TEMP);
                  SPACE(1); ABSFIXT(4,0,A[3]);
                  IF A[5]>1 ∧ A[5]≠100 THEN PRINTTEXT(‹D›) ELSE SPACE(1);
                  DIRECTION(A[3],RICHT); GEMRICHT(VR,RICHT,GEMR); ABSFIXT(3,0,GEMR);
                  ABSFIXT(4,0,A[4]);
                  IF A[5]>99 THEN PRINTTEXT(‹V›) ELSE SPACE(1);
                  VELOCITY(A[4],VGS,SNELH); ABSFIXT(3,1,SNELH);
                  OOST:=SNELH*SIN(GEMR*RAD); FIXT(3,1,OOST);
                  NOORD:=SNELH*COS(GEMR*RAD); FIXT(3.1,NOORD);
                  IF A[5]=1 THEN PRINTTEXT(‹A›) ELSE SPACE(1);
                  IF PONS THEN
                  BEGIN   PONSZUINIG(3,1,OOST); PONSZUINIG(3,1,NOORD);
                          IF A[5]=1 THEN BEGIN  RUNOUT; PUNLCR; PUHEP(LOWER); END;
                          TGW:=TGW+1; IF IMOD(TGW,10)=0 THEN PUNLCR
                  END;
                  SPACE(14); PLOT(GEMR,SNELH); NLCR;
                  IF AANDERAA THEN
                  BEGIN   IF A[5]≠1 THEN BEGIN SDNEXT; SALINITY(GSAL,HSAL); DIEPTE(GDIEP,HDIEP); END;
                          SAL:=SAL+HSAL; DIEP:=DIEP+HDIEP;
                  END;
                  GEMTEMP:=GEMTEMP+TEMP;
                  GEMOOST:=GEMOOST+OOST;
                  GEMNOORD:=GEMNOORD+NOORD;
                  TIJD:=TIJD+V; TIMING(TIJD);
                  IF (ENTIER(TIJD/100 )>S ∧S ≠0) ∨ (TIJD/100<1 ∧ S=23) ∨ (S=0 ∧ TIJD>60) THEN
                  BEGIN   S:=S+1;  PRINTTEXT(‹HOURLY MEAN›); ABSFIXT(2,0,S); UURW[7*K+1]:=DAG; UURW[7*K+2]:=S;
                          UURW[7*K+6]:=GEMOOST/TEL;
                          UURW[7*K+7]:=GEMNOORD/TEL;
                          UURW[7*K+3]:=GEMTEMP/TEL;
                          GEMTEMP:=GEMOOST:=GEMNOORD:=0;
                          IF AANDERAA THEN
                          BEGIN   UURW[7*K+4]:=SAL/TEL; UURW[7*K+5]:=DIEP/TEL; SAL:=DIEP:=0; END;
                          TEL:=0;
                          IF S=24 THEN
                          BEGIN   UURW[7*K+1]:=DAG-1; DAGTEL; S:=0; DTEL:=0;
                                  OUTARRAY(DRUM,UURADR,UURW); HOLD(UURW); UURADR:=UURADR+14*(K+1); TOTUUR:=TOTUUR+K+1; K:=0;
                                  IF PONS THEN BEGIN PUNLCR; RUNOUT; PUHEP(LOWER); END
                          END     ELSE K:=K+1;
                          IF IMOD(S*N,48)=0 THEN
                          BEGIN NEWPAGE; ABSFIXT(8,0,DAG); NLCR; KOP END; NLCR;
                  END;
                  VGT:=A[2]; VGR:=A[3]; VGS:=A[4];
                  TEL:=TEL+1; DTEL:=DTEL+1;
                  GOTO ALLES;
    HALT:  TYD[5]:=TIME; IF PONS THEN BEGIN PUNLCR;RUNOUT;RUNOUT; END;
           TWO:=TRUE; K:=0; TEL:=1; UURADR:=140000; INARRAY(DRUM,UURADR,UURW); UURADR:=UURADR+336; HOLD(UURW);
    PRINT: IF UURW[7*K+1]=VDAG THEN BEGIN NLCR;SPACE(10);
                                    END
           ELSE BEGIN      VDAG:=UURW[7*K+1];
                           IF TWO THEN BEGIN NEWPAGE; UURKOP; TWO:=FALSE; END
                           ELSE TWO:=TRUE;NLCR;
                           ABSFIXT(8,0,VDAG);
                END;
           ABSFIXT(2,0,UURW[7*K+2]); FIXT(2,1,UURW[7*K+3]);
           IF AANDFRAA THEN BEGIN IF UURW[7*K+4]=0 THEN
                                      SPACE(6) ELSE ABSFIXT(2,1,UURW[7*K+4]);
                                      IF UURW[7*K+5]=0 THEN SPACE(6) ELSE
                                      ABSFIXT(2,1,UURW[7*K+5]);
                            END
                            ELSE SPACE(12);
```

```
           FIXT(3,1,UURW[7*K+6]); FIXT(3,1,UURW[7*K+7]);
           TRANSF(UURW[7*K+6],UURW[7*K+7],SNELH,RICHT);
           ABSFIXT(3,0,RICHT);ABSFIXT(3,1,SNELH);
           SPACE(14);PLOT(RICHT,SNELH);
           TEL:=TEL+1;
           IF TEL>TOTUUR THEN GOTO FINISH; K:=K+1;
           IF K=24 THEN BEGIN INARRAY(DRUM,UURADR,UURW); UURADR:=UURADR+336; K:=0; HOLD(UURW);
                        END;
           GOTO PRINT;
           FINISH: END REKENBLOK;
           TYD[6]:=TIME;
           NEWPAGE;
           PRINTTEXT(‹AANTAL WAARNEMINGEN:›); ABSFIXT(6,0,TW); NLCR;
           PRINTTEXT(‹AANTAL GEPONSTE WAARNEMINGEN:›); ABSFIXT(6,0,TGW); NLCR; NLCR;

           PRINTTEXT(‹        OVERDRACHT  ERRORFILL   CORRIGEREN   REKENEN    UURTABEL›); NLCR;
           FOR J:= 1 STEP 1 UNTIL 6 DO
           BEGIN ABSFIXT(6,0,TYD[J]); SPACE(4) END;

           IF ¬COMPL THEN
           BEGIN   COMMENT DIT GEDEELTE ALS DE SERIE-VERWERKING ONDERBROKEN WERD;
                   NEWPAGE;
                   TEND:=TEND+1; TWFIRST:=TWBR[TEND-1]+1; TWG:=TWBR[TEND];
                   IF TWG=TOTAL THEN COMPL:=TRUE;
                   TYD[1]:=TYD[2]:=TYD[3]:=TIME;
                   ONE:=TRUE; I:=0;                          GOTO FILLERROR;
           END RETURN;

    STOP:  NLCR; NLCR; PRINTTEXT(‹EINDE PRAG-DATA -COMPUTATION›);
           IF PONS THEN BEGIN PUNLCR;RUNOUT; FIXP(4,0,1000); PUNLCR;RUNOUT;RUNOUT; END;
    END
    END
```

Appendix F.

ACM (Actual Time Current Meters)

Algol-60 source listing.

```
741007:
BEGIN   COMMENT PRAG-ACTUAL TIME CURRENT METERS-VERSIE;
        REAL RAD,DELTAT,TIJD,VORTIJD,NOORD,OOST,RICHT,MAGN,VORNRD,VOROOST,NOORD1,NOORD2,OOST1,OOST2,
                ETIJD,EUUR,UURHULP,UURTIJD,TENTIJD,FACTOR,HULP1,HULP2,NSOM,OSOM,HNRD,HOOST;
        INTEGER I,J,DAG,EDAG,IHULP,ADRES,ADR,CORADR,TYPE,METNO,LAT,LONG,DEPTH,WDEPTH,DATUM,
                CAMPNR,STATNR,TEL,UURTEL,DAGTEL,AANTAL,REGEL,VERSIE,LOOP,BUFADR,BUFLOOP,NRNEXT:
        BOOLEAN VECTOR,TIEN,BUFBOOL,BLOOP,TWEE:
        REAL ARRAY T*D[1:9],ABUF,BBUF[1:75],WEGN,WEGO[1:100];
        INTEGER ARRAY WIG,WAG[1:200]:
        REAL EINDTIJD,BEGINTIJD; INTEGER BEGINNR,UUR:

        PROCEDURE ADDLASTTERMS;
        BEGIN   NOORDSOM:=NOORDSOM+(NOORD*(EINDTIJD-VORTIJD)*(EINDTIJD-VORTIJD)
                -VORNRD*(TIJD-EINDTIJD)*(TIJD-EINDTIJD))/(2*DELTAT*DELTAT);
                OOSTSOM:=OOSTSOM+(OOST*(EINDTIJD-VORTIJD)*(EINDTIJD-VORTIJD)
                -VOROOST*(TIJD-EINDTIJD)*(TIJD-EINDTIJD))/(2*DELTAT*DELTAT);
                        END;

        PROCEDURE SCALERESULT;
        BEGIN   NOORDSOM:=NOORDSOM*DELTAT/60;
                OOSTSOM:=OOSTSOM*DELTAT/60:
                DIRECTION(RICHT,NOORDSOM,OOSTSOM);
                MAGN:=SQRT(NOORDSOM*NOORDSOM+OOSTSOM*OOSTSOM);
        END;

        PROCEDURE LISTRESULT;
        BEGIN   SPACE(10); ABSFIXT(4,0,UUR); FIXT(3,1,NOORDSOM); FIXT(3,1,OOSTSOM);
                ABSFIXT(3,0,RICHT); ABSFIXT(3,1,MAGN); SPACE(14); PLOT(RICHT,MAGN); NLCR;
        END;

        PROCEDURE UPDATEBEGINANDEND;
        BEGIN   BEGINNR:=NRNEXT-1;
                BEGINTIJD:=EINDTIJD;
                UUR:=UUR+1;
                IF UUR=24 THEN
                BEGIN   UUR:=0; DAG:=DAG+1; DAGTEL;
                        IF TWEE THEN
                        BEGIN NEWPAGE; UURKOP; END:
                        TWEE:=¬TWEE;
                        ABSFIXT(8,0,DAG);
                        NLCR;
                END:
                EINDTIJD:=100*UUR+30
        END;

        PROCEDURE COMPUTEFIRSTTERMS;
        BEGIN   VORNRD:=NOORD; VOROOST:=OOST; VORTIJD:=TIJD;
                GETNEXT;
                IF ¬ENDOFDATAFILE THEN
                BEGIN   NOORDSOM:=(VORNRD*(TIJD-BEGINTIJD)*(TIJD-BEGINTIJD)
                        -NOORD*(BEGINTIJD-VORTIJD)*(BEGINTIJD-VORTIJD))/(2*DELTAT*DELTAT);
                        OOSTSOM:=(VOROOST*(TIJD-BEGINTIJD)*(TIJD-BEGINTIJD)
                        -OOST*(BEGINTIJD-VORTIJD)*(BEGINTIJD-VORTIJD))/(2*DELTAT*DELTAT);
                END:
        END;

        PROCEDURE ADDCOMPONENTS;
        BEGIN   NOORDSOM:=NOORDSOM+NOORD;
                OOSTSUM:=OOSTSOM+OOST;
```

```
                VORNRD:=NOORD; VOROOST:=OOST; VORTIJD:=TIJD;
                GETNEXT;
        END;

        PROCEDURE ADDCOMPONENTS AROUND MIDNIGHT;
        BEGIN   UNTIL(MIDNIGHT)DO:(ADDCOMPONENTS);
                WHILE(TIME IN RANGE)DO:(ADDCOMPONENTS);
        END;
        BOOLEAN PROCEDURE MIDNIGHT;
        MIDNIGHT:=TIJD<VORTIJD∨ENDOFDATAFILE;
        PROCEDURE WHILE(CONDITION)DO:(FUNCTION);
        BOOLEAN PROCEDURE CONDITION; PROCEDURE FUNCTION;
        BEGIN   TEST: IF CONDITION THEN
                BEGIN FUNCTION; GOTO TEST;
                END:
        END;
        REAL NOORDSOM,OOSTSOM;

        BOOLEAN PROCEDURE ENDOFDATAFILE;
        ENDOFDATAFILE:=NRNEXT≥AANTAL;
        PROCEDURE UNTIL(CONDITION)DO:(FUNCTION);
        BOOLEAN PROCEDURE CONDITION; PROCEDURE FUNCTION;
        BEGIN   PERFORM: FUNCTION;
                IF ¬CONDITION THEN GOTO PERFORM;
        END;

        PROCEDURE COMPUTEANDLISTHOURLYMEANS;
        BEGIN   STARTBUF(BEGINNR);
                GETNEXT;
                IF TIJD≤BEGINTIJD-DELTAT THEN
                UNTIL(PROPERTIME)DO:(GETNEXT)
                ELSE IF TIJD>BEGINTIJD THEN
                UNTIL(PROPERTIME)DO:(RETRY);
                COMPUTEFIRSTTERMS;
                IF UUR≠0 THEN WHILE(TIME IN RANGE)DO:(ADDCOMPONENTS)
                ELSE ADDCOMPONENTSAROUNDMIDNIGHT;
                IF ¬ENDOFDATAFILE THEN
                BEGIN
                        ADDLASTTERMS;
                        SCALERESULT;
                        LISTRESULT:
                        UPDATEBEGINANDEND;
                END;
        END;
        BOOLEAN PROCEDURE PROPERTIME;
        PROPERTIME:=BEGINTIJD-DELTAT<TIJD∧TIJD≤BEGINTIJD,
        PROCEDURE RETRY;
        BEGIN   BEGINNR:=BEGINNR-1;
                IF BEGINNR=0 THEN BREEKAF(≮FOUT IN RETRY≯);
                STARTBUF(BEGINNR):
                GETNEXT;
        END;
        BOOLEAN PROCEDURE TIMEINRANGE;
        TIMEINRANGE:=TIJD<EINDTIJD∧¬ENDOFDATAFILE;

        PROCEDURE TIDALMEANS;
        BEGIN   IF ¬ENDOFDATAFILE THEN
                MORNINGPERIOD;
                IF ¬ENDOFDATAFILE THEN
                EVENINGPERIOD;
        END;

        PROCEDURE MORNINGPERIOD;
        BEGIN   BEGINTIJD:=HULP2; EINDTIJD:=1160;
                GETNEXT;
                COMMENT 1160 INSTEAD OF 1200 FOR TESTS;
                IF TIJD≤BEGINTIJD-DELTAT THEN
                UNTIL(PROPERTIME)DO:(GETNEXT)
                ELSE IF TIJD>BEGINTIJD THEN
                UNTIL(PROPERTIME)DO:(RETRY);
                COMPUTEFIRSTTERMS;
                UNTIL(MIDNIGHT)DO:(ADDCOMPONENTS);
                WHILE(TIME IN RANGE)DO:(ADDCOMPONENTS);
                IF ¬ ENDOFDATAFILE THEN
                BEGIN
                        ADDMIDDAYTERMS;
                        SCALETIDALRESULTS;
                        LISTMORNINGPERIOD;
                        STARTBUF(NRNEXT-1);
                END;
        END;

        PROCEDURE EVENINGPERIOD;
        BEGIN   BEGINTIJD:=1160; EINDTIJD:=HULP1;
                COMMENT 1160 INSTEAD OF 1200 FOR TESTS;
                GETNEXT;
                IF TIJD≤BEGINTIJD-DELTAT THEN
                UNTIL(PROPERTIME)DO:(GETNEXT)
                ELSE IF TIJD>BEGINTIJD THEN
                UNTIL(PROPERTIME)DO:(RETRY);
                COMPUTEMIDDAYTERMS;
                UNTIL(MIDNIGHT)DO:(ADDCOMPONENTS);
                WHILE(TIME IN RANGE)DO:(ADDCOMPONENTS);
                IF ¬ ENDOFDATAFILE THEN
                BEGIN
                        ADDLASTTERMS;
                        SCALETIDALRESULTS;
                        LISTEVENINGPERIOD;
                        STOREDAILYMEANS;
                        STARTBUF(NRNEXT-CYCLESPERHOUR);
                END;
        END;
        INTEGER CYCLESPERHOUR;

        PROCEDURE SCALETIDALRESULTS;
        BEGIN   NOORDSOM:=NOORDSOM*DELTAT;
                OOSTSOM:=OOSTSOM*DELTAT;
                NOORD1:=NOORD1+NOORDSOM;
                OOST1:=OOST1+OOSTSOM;
                NOORDSOM:=NOORDSOM/745.2;
                OOSTSOM:=OOSTSOM/745.2;
                DIRECTION(RICHT,NOORDSOM,OOSTSOM);
                MAGN:=SQRT(NOORDSOM*NOORDSOM+OOSTSOM*OOSTSOM);
        END;

        PROCEDURE ADDMIDDAYTERMS;
        BEGIN   NOORDSOM:=NOORDSOM+(NOORD*(1160-VORTIJD)*(1160-VORTIJD)
                -VORNRD*(1200-TIJD)*(1200-TIJD))/(2*DELTAT*DELTAT);
                OOSTSOM:=OOSTSOM+(OOST*(1160-VORTIJD)*(1160-VORTIJD)
                -VOROOST*(1200-TIJD)*(1200-TIJD))/(2*DELTAT*DELTAT);
```

```
        END;

        PROCEDURE COMPUTEMIDDAYTERMS;
        BEGIN   VORNRD:=NOORD; VOROOST:=OOST;
                VORTIJD:=TIJD;
                GETNEXT;
                IF ¬ ENDOFDATAFILE THEN
                BEGIN   NOORDSOM:=(VORNRD*(TIJD-1200)*(TIJD-1200)
                        -NOORD*(1160-VORTIJD)*(1160-VORTIJD))/(2*DELTAT*DELTAT);
                        OOSTSOM:=(VOROOST*(TIJD-1200)*(TIJD-1200)
                        -OOST*(1160-VORTIJD)*(1160-VORTIJD))/(2*DELTAT*DELTAT);
                END;
        END;

        PROCEDURE LISTMORNINGPERIOD;
        BEGIN   ABSFIXT(8,0,DAG);
                FIXT(3,1,NOORDSOM); FIXT(3,1,OOSTSOM);
                ABSFIXT(3,0,RICHT); ABSFIXT(3,1,MAGN);
        END;

        PROCEDURE LISTEVENINGPERIOD;
        BEGIN   FIXT(3,1,NOORDSOM); FIXT(3,1,OOSTSOM);
                ABSFIXT(3,0,RICHT); ABSFIXT(3,1,MAGN);
        END;

        PROCEDURE STOREDAILYMEANS;
        BEGIN   J:=J+1;
                WEGN[J]:=NOORD1*144*⏨-4/24.84;
                WEGO[J]:=OOST1*144*⏨-4/24.84;
                NOORD1:=NOORD1/1490.4;
                OOST1:=OOST1/1490.4;
                DIRECTION(RICHT,NOORD1,OOST1);
                MAGN:=SQRT(NOORD1*NOORD1+OOST1*OOST1);
                FIXT(3,1,NOORD1); FIXT(3,1,OOST1);
                ABSFIXT(3,0,RICHT); ABSFIXT(3,1,MAGN);
                NOORD1:=OOST1:=0;
                REGEL:=REGEL+1;
                IF REGEL>54 THEN
                BEGIN   NEWPAGE;
                        DAGKOP;
                        REGEL:=1;
                END
                ELSE NLCR;
                DAG:=DAG+1; DAGTEL;
                STARTBUF(NRNEXT-CYCLESPERHOUR-1);
        END;

        INTEGER PROCEDURE IMOD(INT,GETAL); VALUE INT,GETAL; INTEGER INT,GETAL;
        BEGIN   IMOD:=INT-INT÷GETAL*GETAL END;

        REAL PROCEDURE RMOD(REAL,GETAL); VALUE REAL,GETAL; REAL REAL,GETAL;
        BEGIN   RMOD:=REAL-ENTIER(REAL/GETAL)*GETAL END;

        PROCEDURE TIMING(TIJD); REAL TIJD;
        BEGIN   IF RMOD(TIJD,100)≥60 THEN TIJD:=TIJD+40;
                IF TIJD≥2400 THEN BEGIN TIJD:=TIJD-2400; DAG:=DAG+1 END;
        END TIMING;

        PROCEDURE DAGTEL;
        BEGIN   INTEGER HDAG,MND;
        HDAG:=IMOD(DAG,100);
        IF HDAG≤28 THEN GOTO DAGKLAAR;
        MND:=IMOD((DAG-HDAG)÷100,100); IF MND=2 THEN
        BEGIN   IF IMOD(DAG÷10000,4)=0 THEN
                BEGIN   IF HDAG=30 THEN DAG:=DAG+71; GOTO DAGKLAAR END
                ELSE BEGIN DAG:=DAG+72; GOTO DAGKLAAR; END;
        END MND=2;
        IF MND=1 ∨ MND=3 ∨ MND=5 ∨ MND=7 ∨ MND=8 ∨ MND=10 ∨ MND=12 THEN
        BEGIN   IF HDAG=32 THEN
                BEGIN   DAG:=DAG+69; IF MND=12 THEN DAG:=DAG+8800; END:
                GOTO DAGKLAAR;
        END:
        IF HDAG=31 THEN DAG:=DAG+70;
DAGKLAAR:
END DAGTEL:

PROCEDURE DIRECTION(RICHT,NOORD,OOST); VALUE NOORD,OOST; REAL RICHT,NOORD,OOST:
BEGIN   REAL HULP;
        IF ABS(NOORD)<0.1 THEN
        BEGIN   RICHT:= IF OOST>0 THEN 90 ELSE 270;
                GOTO UITDIR;
        END;
        IF ABS(OOST)<0.1 THEN
        BEGIN   RICHT:= IF NOORD>0 THEN 360 ELSE 180;
                GOTO UITDIR;
        END;
        HULP:=ABS(OOST/NOORD);
        HULP:=ARCTAN(HULP)/RAD:
        IF NOORD>0 THEN RICHT:= IF OOST>0 THEN HULP ELSE (360-HULP)
        ELSE RICHT:= IF OOST>0 THEN (180-HULP) ELSE (180+HULP);
UITDIR:
END DIRECTION;

PROCEDURE PLOT(RICHT,MAGN); VALUE RICHT,MAGN; REAL RICHT,MAGN;
BEGIN   INTEGER SP1,SP2;
        SP1:=RICHT/10; SP2:=MAGN/8;
        IF SP2>SP1 THEN
        BEGIN   SPACE(SP1); PRINTTEXT({+});
                IF SP2≤64 THEN BEGIN SPACE(SP2-SP1-1); PRINTTEXT({.}): END ELSE
                BEGIN SPACE(64-SP1-1); PRINTTEXT({∨}); END
        END
        ELSE IF SP2<SP1 THEN BEGIN SPACE(SP2); PRINTTEXT({.}); SPACE(SP1-SP2-1); PRINTTEXT({+}); END
        ELSE BEGIN SPACE(SP1); PRINTTEXT({S}); END;
END PLOT;

PROCEDURE WIGWAG;
BEGIN
LWIG:   FOR I:= 1 STEP 1 UNTIL 200 DO
        BEGIN   WIG[I]:=10*READ;
                IF WIG[I]=10000 THEN
                BEGIN   HOLD(WAG); OUTARRAY(DRUM,ADRES,WIG); HOLD(WIG): GOTO WWKLAAR; END;
        END:
        HOLD(WAG);
        OUTARRAY(DRUM,ADRES,WIG); ADRES:=ADRES+200;
LWAG:   FOR I:= 1 STEP 1 UNTIL 200 DO
        BEGIN   WAG[I]:=10*READ;
                IF WAG[I]=10000 THEN BEGIN HOLD(WIG); OUTARRAY(DRUM,ADRES,WAG): HOLD(WAG); GOTO WWKLAAR: END;
        END;
        HOLD(WIG);
        OUTARRAY(DRUM,ADRES,WAG); ADRES:=ADRES+200; GOTO LWIG;
```

```
WWKLAAR:
END WIGWAG;

PROCEDURE TENKOP;
BEGIN   PRINTTEXT({TEN MINUTE VALUES}); NLCR:
        PRINTTEXT({   DATE     TIME  NORTH  EAST   DIR  VELOC}); NLCR;
        PRINTTEXT({ YYYYMMDD   GMT    COMP   COMP  DEGR  CM/SEC});
        NLCR;NLCR;
END TENKOP;

PROCEDURE UURKOP;
BEGIN   PRINTTEXT({HOURLY MEANS}); NLCR;
        PRINTTEXT({   DATE     TIME  NORTH  EAST   DIR  VELOC}); SPACE(15):
        PRINTTEXT({+0        +90       +180      +270      +360                    DEGR}); NLCR;
        PRINTTEXT({ YYYYMMDD   GMT    COMP   COMP  DEGR  CM/SEC}); SPACE(14);
        PRINTTEXT({.0        .20       .40       .60       .80       .100          CM/S});
        NLCR;NLCR;
END UURKOP;

PROCEDURE DAGKOP;
BEGIN   PRINTTEXT({TIDAL MEANS   24.84 HOURS MEANS}); NLCR:
        PRINTTEXT({   DATE}); SPACE(9):
        PRINTTEXT({MORNING PERIOD}); SPACE(12);
        PRINTTEXT({EVENING PERIOD}); SPACE(13);
        PRINTTEXT({FULL PERIOD}); NLCR;
        PRINTTEXT({ YYYYMMDD  NORTH  EAST   DIR  VELOC  NORTH  EAST   DIR  VELOC  NORTH  EAST   DIR  VELOC}): NLCR:
        SPACE(12);
        PRINTTEXT({COMP  COMP  DEGR  CM/SEC  COMP  COMP  DEGR  CM/SEC  COMP  COMP  DEGR  CM/SEC}): NLCR;NLCR:
END DAGKOP:

PROCEDURE POETS;
BEGIN   INTEGER HADRES,K;
        INTEGER ARRAY M[1:2];
        NEWPAGE; PRINTTEXT({CORRECTED DATA});
        NLCR: PRINTTEXT({NUMBER  NORTH    EAST});
        NLCR; NLCR;
WPOETS: K:=READ;
        IF K=-2 THEN GOTO EINDPOETS;
        HADRES:=ADRES+2*K-2;
        M[1]:=10 * READ; M[2]:=10 * READ;
        HOLD(M); OUTARRAY(DRUM,HADRES,M):
        ABSFIXT(5,0,K); FIXT(3,1,M[2]/10);
                        FIXT(3,1,M[1]/10); NLCR;
        GOTO    WPOETS;
EINDPOETS: HOLD(M);
END     POETS;


PROCEDURE KAP(N,M); INTEGER N,M;
BEGIN   INTEGER K;
        IF N≠0 THEN
        BEGIN   ADRES:=ADRES+2 * N;
                AANTAL:=AANTAL-N;
                FOR K:=1, K+1 WHILE K≤N DO
                BEGIN   ETIJD:=ETIJD+DELTAT;
                        TIMING(ETIJD);
                        DAGTEL:
                END:
                PRINTTEXT({THE FIRST});
                ABSFIXT(4,0,N);
                PRINTTEXT(‹DATA-POINTS HAVE BEEN SKIPPED›);
                NLCR; NLCR;
                PRINTTEXT(‹THE REVISED TIME OF FIRST MEASUREMENT IS GMT:›);
                ABSFIXT(3,0,DAG); ABSFIXT(4,0,ETIJD); NLCR; NLCR;
        END;

        IF M≠0 THEN
        BEGIN   AANTAL:=AANTAL-M;
                PRINTTEXT(‹THE LAST›);
                ABSFIXT(4,0,M);
                PRINTTEXT(‹DATA-POINTS HAVE BEEN SKIPPED›);
        END;
END     KAP;

PROCEDURE GETNEXT;
BEGIN   NRNEXT:=NRNEXT+1; IF BUFBOOL THEN GOTO GETTW;
        TIJD:=ABUF[BUFLOOP+1];
        NOORD:=ABUF[BUFLOOP+2];
        OOST:=ABUF[BUFLOOP+3];
        BUFLOOP:=BUFLOOP+3; IF BUFLOOP=75 THEN
        BEGIN   IF BUFADR≥90000 THEN BREEKAF(‹ADRESFOUT IN GETNEXT›);
                INARRAY(DRUM,BUFADR,ABUF); BUFADR:=BUFADR+150;
                BUFLOOP:=0; BUFBOOL:=TRUE;
                HOLD(BBUF);
        END;
        GOTO ENDGET;
GETTW:  TIJD:=BBUF[BUFLOOP+1]; NOORD:=BBUF[BUFLOOP+2];
        OOST:=BBUF[BUFLOOP+3]; BUFLOOP:=BUFLOOP+3;
        IF BUFLOOP=75 THEN
        BEGIN   IF BUFADR≥90000 THEN BREEKAF(‹ADRESFOUT IN GETNEXT›);
                INARRAY(DRUM,BUFADR,BBUF); BUFADR:=BUFADR+150;
                BUFLOOP:=0; BUFBOOL:=FALSE;
                HOLD(ABUF);
        END;
ENDGET:
END GETNEXT;

PROCEDURE BREEKAF(TEXT); STRING TEXT;
BEGIN   CARRIAGE(4); PRINTTEXT(TEXT);
        CARRIAGE(4); GOTO STOP;
END;

PROCEDURE PUTNEXT;
BEGIN   IF BUFBOOL THEN GOTO PUTTW;
        ABUF[BUFLOOP+1]:=TIJD;
        ABUF[BUFLOOP+2]:=NOORD;
        ABUF[BUFLOOP+3]:=OOST;
        BUFLOOP:=BUFLOOP+3;
        IF BUFLOOP=75 THEN
        BEGIN   IF BUFADR≥90000 THEN BREEKAF(‹ADRESFOUT IN PUTNEXT›);
                OUTARRAY(DRUM,BUFADR,ABUF);
                BUFADR:=BUFADR+150;
                BUFLOOP:=0; BUFBOOL:=TRUE;
                HOLD(BBUF);
        END;
        GOTO ENDPUT;
PUTTW:  BBUF[BUFLOOP+1]:=TIJD;
        BBUF[BUFLOOP+2]:=NOORD;
        BBUF[BUFLOOP+3]:=OOST;
        BUFLOOP:=BUFLOOP+3;
```

```
        IF BUFLOOP=75 THEN
        BEGIN   IF BUFADR≥90000 THEN BREEKAF(‹ADRESFOUT IN PUTNEXT›);
                OUTARRAY(DRUM,BUFADR,BBUF);
                BUFADR:=BUFADR+150; BUFLOOP:=0; BUFBOOL:=FALSE;
                HOLD(ABUF);
        END;
ENDPUT: NRNEXT:=NRNEXT+1;
END PUTNEXT;
PROCEDURE CONVERT;
BEGIN   READNEXT;
        NOORD:=NOORD*FACTOR;
        OOST:=OOST*FACTOR;
        FIXP(3,1,NOORD);FIXP(3,1,OOST);
        SPACE(5);
        PUTNEXT;
        IF IMOD(NRNEXT,5)=0 THEN PUNLCR;
        IF ENDOFDATAFILE THEN
        BEGIN   IF BUFBOOL THEN
                OUTARRAY(DRUM,BUFADR,BBUF)
                ELSE
                OUTARRAY(DRUM,BUFADR,ABUF);
                HOLD(ABUF);HOLD(BBUF);
                PUNLCR;
                RUNOUT; RUNOUT;
        END
        ELSE    BEGIN   TIJD:=TIJD+DELTAT;
                        TIMING(TIJD);
                END;
END;
BOOLEAN PROCEDURE TENTIMEINRANGE;
TENTIMEINRANGE:=TENTIJD≥VORTIJD;
PROCEDURE FINDSTART;
BEGIN   IF TENTIJD=VORTIJD THEN
        BEGIN   HNRD:=NOORD; HOOST:=OOST;
                PRINTTEN;
        END
        ELSE IF TENTIJD<VORTIJD THEN
        BEGIN   TENTIJD:=TENTIJD+10;
                TIMING(TENTIJD);
                DAGTEL;
        END;
END;
PROCEDURE PRINTTEN;
BEGIN   DIRECTION(RICHT,HNRD,HOOST);
        MAGN:=SQRT(HNRD*HNRD+HOOST*HOOST);
        IF TENTIJD=0∨REGEL=4
        THEN ABSFIXT(8,0,DAG)
        ELSE SPACE(10);
        ABSFIXT(4,0,TENTIJD);
        FIXT(3,1,HNRD);FIXT(3,1,HOOST);
        ABSFIXT(3,0,RICHT);ABSFIXT(3,1,MAGN);
        REGEL:=REGEL+1;
        IF REGEL=60 THEN        BEGIN   NEWPAGE;
                                        TENKOP;
                                        REGEL:=4;
                                END
                    ELSE        NLCR;
        TENTIJD:=TENTIJD+10; TIMING(TENTIJD);
        DAGTEL;
END;
```

```
          PROCEDURE LISTTENMINUTES;
          BEGIN   GETNEXT;
                  IF TENTIJD=0 THEN
                  BEGIN   IF TIJD>DELTAT THEN UPDATETEN
                          ELSE    BEGIN   VORTIJD:=VORTIJD-2360;
                                          INTERPOLATIE;
                                          PRINTTEN;
                                  END;
                  END
                  ELSE IF TENTIJD>TIJD THEN UPDATETEN ELSE
                  BEGIN INTERPOLATIE; PRINTTEN; END;
          END;
          PROCEDURE UPDATETEN;
          BEGIN   VORNRD:=NOORD; VOROOST:=OOST; VORTIJD:=TIJD; END;
          PROCEDURE INTERPOLATIE;
          BEGIN   IF TENTIJD-VORTIJD>40 THEN
                  VORTIJD:=VORTIJD+40;
                  HNRD:=((TIJD-TENTIJD)*VORNRD+(TENTIJD-VORTIJD)*NOORD)/DELTAT;
                  HOOST:=((TIJD-TENTIJD)*VOROOST+(TENTIJD-VORTIJD)*NOORD)/DELTAT;
          END;

          PROCEDURE READNEXT;
          BEGIN   IF BLOOP THEN GOTO RDTW;
                  NOORD:=WIG[LOOP+2]; OOST:=WIG[LOOP+1];
                  LOOP:=LOOP+2;
                  IF LOOP=200 THEN
                  BEGIN   IF ADRES≥130000 THEN BREEKAF(¢ADRESFOUT IN READNEXT$);
                          INARRAY(DRUM,ADRES,WIG); ADRES:=ADRES+200;
                          LOOP:=0; BLOOP:=TRUE; HOLD(WAG);
                  END;
                  GOTO ENDRDNXT;
          RDTW:   NOORD:=WAG[LOOP+2]; OOST:=WAG[LOOP+1]; LOOP:=LOOP+2;
                  IF LOOP=200 THEN
                  BEGIN   IF ADRES≥130000 THEN BREEKAF(¢ADRESFOUT IN READNEXT$);
                          INARRAY(DRUM,ADRES,WAG); ADRES:=ADRES+200;
                          LOOP:=0; BLOOP:=FALSE;
                          HOLD(WIG);
                  END;
          ENDRDNXT:
          END READNEXT;

          PROCEDURE STARTBUF(FIRST); VALUE FIRST; INTEGER FIRST;
          BEGIN   BUFADR:=6*FIRST-6; NRNEXT:=FIRST-1;
                  IF BUFADR≥90000 THEN BREEKAF(¢ADRESFOUT IN STARTBUF$);
                  INARRAY(DRUM,BUFADR,ABUF); BUFADR:=BUFADR+150;
                  INARRAY(DRUM,BUFADR,BBUF); BUFADR:=BUFADR+150;
                  BUFLOOP:=0; BUFBOOL:=FALSE; HOLD(ABUF);
          END STARTBUF;

          VERSIE:=741007;

 PROGRAMMA:       TYD[1]:=TIME; TYD[8]:=TYD[9]:=0;
          PRINTTEXT(¢PRAG-ACTUAL TIME CURRENT METERS$); SPACE(70);
          PRINTTEXT(¢VERSIE:$); ABSFIXT(6,0,VERSIE); CARRIAGE(6);
          PRINTTEXT(¢K.N.M.I.       DE BILT        NETHERLANDS$); CARRIAGE(6);
          PRINTTEXT(¢CURRENT METER CAMPAIGN:$); CAMPNR:=READ; STATNR:=READ; ABSFIXT(4,0,CAMPNR);
          PRINTTEXT(¢       ,STATION/PERIODE: $); ABSFIXT(6,0,STATNR); CARRIAGE(4);
          LAT:=READ; LONG:=READ; DEPTH:=READ; SPACE(13);
          PRINTTEXT(¢DEGR  MIN       DEGR  MIN$); NLCR;
          PRINTTEXT(¢POSITION:$); ABSFIXT(6,0,LAT÷100); ABSFIXT(4,0,IMOD(LAT,100));
```

```
          IF LAT>0 THEN PRINTTEXT(¢N$) ELSE PRINTTEXT(¢S$);
          ABSFIXT(6,0,LONG÷100); ABSFIXT(4,0,IMOD(LONG,100));
          IF LONG>0 THEN PRINTTEXT(¢E$) ELSE PRINTTEXT(¢W$); NLCR; NLCR;
          PRINTTEXT(¢INSTRUMENT DEPTH IN METERS:$); ABSFIXT(4,0,DEPTH); NLCR; NLCR;
          WDEPTH:=READ; METNO:=READ;
          PRINTTEXT(¢WATER DEPTH IN METERS:$); ABSFIXT(4,0,WDEPTH); NLCR; NLCR;
          PRINTTEXT(¢NUMBER OF INSTRUMENT:$); ABSFIXT(3,0,METNO); NLCR; NLCR;
          DATUM:=READ; ETIJD:=READ; DELTAT:=READ;
          I:=READ; IF I=0 THEN TIEN:=FALSE ELSE
          IF I=1 THEN TIEN:=TRUE ELSE BREEKAF(¢FOUT IN VOORLOOPBAND$);
          VECTOR:=TRUE;
          PRINTTEXT(¢TIME OF FIRST MEASUREMENT GMT:$); ABSFIXT(8,0,DATUM); ABSFIXT(4,0,ETIJD); NLCR; NLCR;
          PRINTTEXT(¢TIME INTERVAL IN MINUTES:$); ABSFIXT(2,4,DELTAT);
          IF CAMPNR≠READ ∨ STATNR≠READ THEN BREEKAF(¢IDENTIFICATIE IN G-BAND ONJUIST$);
          TYD[2]:=TIME; ADRES:=100000; WIGWAG;
          TYD[3]:=TIME; AANTAL:=(ADRES-100000+I-1)÷2; TEL:=0; ADRES:=100000;
          TELETEXT(¢WILT U HET CORRECTIE-BANDJE G2 VAN PRAG-ACM INLEGGEN?$);

          IHULP:=READ; DAG:=DATUM;
          IF IHULP=-1 THEN POETS;
          IF IHULP=-1 THEN I:=READ ELSE I:=IHULP;
          J:=READ;
          IF I+J≠0 THEN
          BEGIN   NEWPAGE; KAP(I,J); DATUM:=DAG; END;
          TIJD:=ETIJD; FACTOR:=1/DELTAT;
          EDAG:=DATUM;
          IF DELTAT -4<5 THEN FACTOR:=FACTOR/2;

          RUNOUT; RUNOUT; PUNLCR;
          PUTEXT(¢ CORRECTED DATA OF CAMPAIGN$); ABSFIXP(4,0,CAMPNR);
          PUTEXT(¢ STATION $); ABSFIXP(4,0,STATNR); PUNLCR;
          PUTEXT(¢ NORTH  EAST$); PUNLCR;
          RUNOUT; RAD:=ARCTAN(1)/45; TYD[4]:=TIME;
          INARRAY(DRUM,ADRES,WIG); ADRES:=ADRES+200;
          INARRAY(DRUM,ADRES,WAG); ADRES:=ADRES+200;
          LOOP:=0; BLOOP:=FALSE; BUFADR:=0; BUFBOOL:=FALSE; BUFLOOP:=0;
          HOLD(WIG);

          OMZET:  NRNEXT:=0;
                  UNTIL(ENDOFDATAFILE)DO:(CONVERT);
                  TYD[5]:=TIME;
                  IF TIEN THEN
                  BEGIN   STARTBUF(1);
                          DAG:=EDAG:=DATUM;
                          GETNEXT;
                          VORTIJD:=TIJD;VORNRD:=NOORD;
                          VOROOST:=OOST;
                          NEWPAGE; TENKOP;
                          REGEL:=4;
                          TENTIJD:=10*ENTIER(ETIJD/10);
                          UNTIL(TENTIMEINRANGE)DO:
                                  (FINDSTART);
                          UNTIL(ENDOFDATAFILE)DO:
                                  (LISTTENMINUTES);
                  END;

 UURSTART:        NEWPAGE; UURKOP; ABSFIXT(8,0,EDAG); NLCR;
          TYD[6]:=TIME; DAG:=EDAG; TWEE:=FALSE;
                  UUR:=ENTIER(ETIJD/100)+1; DAG:=EDAG;
          BEGINTIJD:=100*UUR-70;
```

```
        IF ETIJD>BEGINTIJD THEN BEGIN   UUR:=UUR+1; BEGINTIJD:=100*UUR-70; END;
        IF UUR =24 THEN
        BEGIN   UUR:=0; BEGINTIJD:=2330; DAG:=DAG+1; DAGTEL;
        END;
        EINDTIJD:=100*UUR+30;
        BEGINNR:=1;
UURLOOP:        UNTIL(ENDOFDATAFILE)DO:(COMPUTE AND LIST HOURLY MEANS);
TIDAL:  TYD[7]:=TIME;
        HULP1:=25.20; HULP2:=2334.80;
        DAG:=EDAG: REGEL:=1; OOST1:=NOORD1:=0; J:=0;
        NEWPAGE; DAGKOP; STARTBUF(1);
        CYCLESPERHOUR:=60/DELTAT;
        IF ETIJD>1200 THEN
        BEGIN   DAG:=DAG+1; DAGTEL; EDAG:=DAG;
                IF ETIJD>2334.80 THEN
                BEGIN   STARTBUF(30/DELTAT+1);
                        ABSFIXT(8,0,DAG);
                        SPACE(26);
                        EVENINGPERIOD;
                END;
        END;
        UNTIL(ENDOFDATAFILE)DO:(TIDALMEANS);
SLOT:   TYD[8]:=TIME;
        DAG:=EDAG; HULP1:=HULP2:=0; NEWPAGE;
        PRINTTEXT(≮PROGRESSIVE VECTOR DIAGRAM DATA≯); CARRIAGE(4);
        PRINTTEXT(≮   DATA       DISTANCE  TRAVELLED PROGRESSIVE≯); NLCR;
        PRINTTEXT(≮ YYYYMMDD   NORTH    EAST   DIR    DIST      N   .    E≯); NLCR;
        PRINTTEXT(≮                COMP     COMP   DEGR    KM≯); NLCR; NLCR;
        FOR I:= 1 STEP 1 UNTIL J DO
        BEGIN   ABSFIXT(8,0,DAG); NOORD:=WEGN[I]; OOST:=WEGO[I];
                HULP1:=HULP1+NOORD; HULP2:=HULP2+OOST; DIRECTION(RICHT,NOORD,OOST);
                MAGN:=SQRT(NOORD*NOORD+OOST*OOST);
                FIXT(3,2,NOORD); FIXT(3,2,OOST); ABSFIXT(4,0,RICHT); ABSFIXT(3,2,MAGN);
                FIXT(3,2,HULP1); FIXT(3,2,HULP2); NLCR;
                DAG:=DAG+1; DAGTEL;
        END;
        TYD[9]:=TIME;
        NEWPAGE;
        PRINTTEXT(≮    START     INLEES    WIGWAG  OMZET/PONS  TENMIN   UURGEM    TIJGEM       TABEL≯);
        NLCR; ABSFIXT(2,0,TYD[1]);
        FOR I:=2 STEP 1 UNTIL 9 DO ABSFIXT(8,0,TYD[I]);
STOP:   CARRIAGE(4);
        PRINTTEXT(≮EINDE PRAG ACTUAL TIME CURRENT METERS≯);
        NLCR; ABSFIXT(6,0,TIME);
END
```