BERICHTE
aus dem
INSTITUT FÜR MEERESKUNDE
an der
CHRISTIAN-ALBRECHTS-UNIVERSITÄT KIEL

Nr. 222

Principles of RAFOS technology at the Institut für Meereskunde Kiel

by

Holger König and Walter Zenk

DOI 10.3289/IFM_BER_222

1992

ISSN 0341–8561

Die "Berichte aus dem Institut für Meereskunde" erscheinen in unregelmäßiger Folge und sind gedacht als Arbeitsunterlagen für den sich mit dem jeweiligen Thema befassenden Personenkreis. Die Hefte werden fortlaufend numeriert. Sie sind unredigierte Beiträge und geben allein die Meinung des Verfassers wieder.

Kopien dieser Arbeit können bezogen werden von:
Institut für Meereskunde
Abt. Meeresphysik
2300 Kiel 1
Düsternbrooker Weg 20

Contents page

Abstract

The term "RAFOS technology" stands for a hydro-acoustic method to observe ocean currents in three dimensions. The system utilises the travel time of coded sound signals between fixed sound sources and drifting receivers. Neutrally buoyant "RAFOS" floats serve as receivers. About ten years ago the RAFOS technology was invented by Tom Rossby at the University of Rhode Island. It was transferred to Kiel where a new float group began its work in 1989. This technical report documents a variety of RAFOS aspects, including float modules, their construction, ballasting, operations and some spin-off instruments. Besides its function as a manual for IfM users in Kiel, it was written as a general source of information for other float groups. Comments on the contents of thus manual would be most welcome.

page

List of Tables

page

List of Figures

page

Principles of RAFOS Technology at the Institut für Meereskunde in Kiel

1 Introduction

The term "RAFOS technology" stands for a hydro-acoustic method for observing ocean currents in three dimensions. The system utilises the travel time of coded sound signals between fixed sound sources and drifting receivers. Neutrally buoyant "RAFOS" floats serve as receivers. They are deployed for labeling moving water masses, i.e. they act like tracers that are advected with ocean currents. Floats are presently being used in Kiel to observe the spreading of the Mediterranean Water tongue in the Atlantic.

The detection of warm and salty lenses of Mediterranean Water in the Iberian Basin (Armi and Zenk, 1984) requires a new spreading and mixing concept for this characteristic water mass in the North Atlantic. Meddies, as the lens-like eddies are called, act like randomly distributed and slowly drifting and decaying salt sources in the ocean. Due to the transient nature of Meddies, long-term current observations at fixed locations (Euleran method) can only provide limited insight into their dynamics. Hence, floating or drifting instruments (Lagrangian method) are much more adequate for tracing distinctive water masses and eddy-like features than moored equipment.

In 1987 the Institut für Meereskunde Kiel (IfM) considered the introduction of a Lagrangian concept for long current observations in the ocean interior. After an initial phase of two years the RAFOS technology (Rossby, 1986) became available from the University of Rhode Island (URI) and was transferred to Kiel (Zenk, 1990). Although at that time this Lagrangian method was not yet commercially available, we got early access to the technology with the kind permission and help of T. Rossby, who had developed it over several years. Financially the introduction was made possible by a grant of the Deutsche Forschungsgemeinschaft as part of the Warm Water Sphere project (SFB 133).

The Institut für Meereskunde was well prepared for the introduction of the RAFOS technology as a new Lagrangian method through manyyears experience with surface drifters. These instruments have been used constantly since the

beginning of the SFB program in 1980 (i.e. Käse and Krauß, 1984; Krauß, 1986). Their decisive success is due to:

* high reliability of the drifters,

* the possibility to measure currents with high accurracy,

* the easy handling of the drifters on board research vessels, and

* the professional tracking service provided by the French company CLS (Collecte Localisation Satellites), Systeme ARGOS.

The introduction of the proven RAFOS technology offered us the potential to transfer a Lagrangian method into the ocean interior and to benefit from the obvious advantages of surface drifter observations. In 1989 a (RAFOS) float group was set up. Within the first three years members of the float group have participated in not less then nine sea-going activities (Table 1.1). During the initial phase we transferred the technology from Rhode Island, adapted it to our requirements, and made it part of the infrastructure of our institute. Extensive testing was necessary: auxiliary devices and methods were developed until we were able to build and ballast RAFOS floats in small production series.

This "Bericht aus dem Institut für Meereskunde" documents the technical results of the introduction of floats in Kiel. It should serve two purposes. Firstly, it should be a source of information for other groups using floats, from which we solicit an intensive feedback. Secondly, we present a technical document (manual), which may serve as a reference not only for the float group in Kiel. It includes technical descriptions, material on shore-based preparations, on sea-borne operations, and on post-mission data collection. Our report represents the status of RAFOS float technology in Kiel until late 1991.

Time	Ship Cruise	Research activities in the NE Atlantic
Dec 88 - Jan 89	METEOR 9/1 BARLAVENTO	CTD, mooring recovery, observations of profiles of sound velocity
May/June 89	POSEIDON 159/2	Deployment of 3 sound sources, CTDO2, XBT, cooperation with freon group, first float tests
Oct/Nov 89	METEOR 11/2	Control observations of sound source array, float tests
March 90	POSEIDON 172/2	Deployment of first MAFOS mooring
May 90	POSEIDON 172/5	Recovery MAFOS and W moorings, float tests, disruption of research programme due to engine problems of POSEIDON
Aug 90	POSEIDON 173/3	Cruise of opportunity, float deployments south of Island, no sound source array available
Oct 90	METEOR 14/1	Float and MAFOS deployments, control observations of sound source array
May 91	POSEIDON 182/4	Completition of sound source array, CTD, XBT, float seeding in cooperation with R. Käse's team
Nov 91	SONNE Test cruise	Float deployment
Jan 92*	POSEIDON 191/1	Exchange of mooring A, float deployments
May 92*	LE SUROIT France	Replacement of mooring W, intercomparison of float types

Table 1.1: *Sea-going activities of the Kiel float group during the first 3 1/2 years of its existence (SFB 1991).*

*After the completion of this technical report

Section 2 briefly reviews the general principles of hydro-acoustically tracked floats that led to the development of RAFOS floats. In section 3 we describe their mechanics, electronic modules, and the controlling software. Float assembly, calibration, and ballasting in Kiel are subjects of Section 4. Section 5 is meant as an operational example. After a short summary of the sound source mooring technique in Section 6, the final two Sections 7 and 8 give an overview of spin-off instruments of the RAFOS technology in Kiel and a flow chart of the subsequent data processing.

2 Observations with Neutrally Buoyant Floats

The history of oceanographic floats dates back to the early fifties. The detection and the application of underwater acoustics for communication during World War II had prepared the field for remote tracking of objects under the sea surface. Early ideas to study the deep water circulation by this method were developed in the USA and in the UK by Stommel (1955) and by Swallow (1955). The first neutrally buoyant floats, known as Swallow floats, "were made from old lengths of scaffolding". They consisted of two parallel tubes, containing the necessary electronics (which operated on 360 DCV!), a battery package and the sound transducer. A whole ship was necessary for tracking individual Swallow floats which - in the beginning - could be heard over only 5 - 6 km distance. Later the fixing or relocation problem was substantially improved by lowering the transmitter frequency from 12 KHz to a few hundred Hertz.

Today Richardson (1991) reports transmission distances of "a few thousand kilometers" for Sound Fixing and Ranging-floats (SOFAR). They operate at 250 Hz. SOFAR floats have been used in larger quantities and with great success since the seventies. Originally their positions were determined by the travel time of sound signals between drifting floats and shore-based (military) listening stations. It is unneccessary to say that for this purpose clocks in the floats and at the receiving stations have to be synchronized within a fraction of a second. Today autonomous listening stations, which can be moored at freely selected sites, have replaced shore-based listening stations. By 1991 American scientists had obtained over 240 SOFAR float-years of data in the North Atlantic at depths from 700 to 2,000 meters (Richardson, 1991).

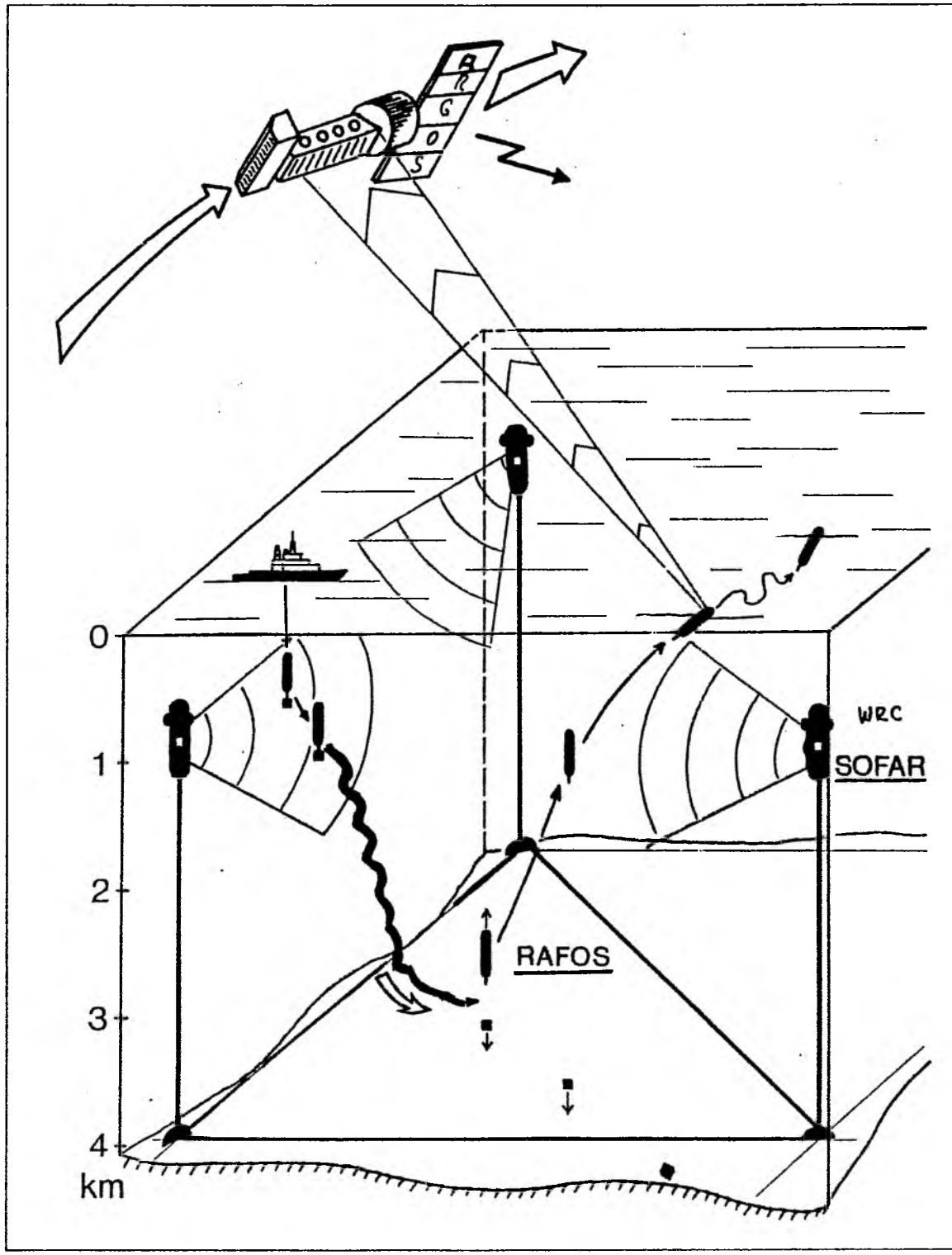


Fig. 2.1 The RAFOS system consists of three major components. Neutrally buoyant RAFOS floats drift in the ocean. They record their position by listening to moored sound sources, which are derived from SOFAR buoys. A satellite link transmits the data once floats have returned to the surface.

When we first considered Lagrangian observations in the Mediterranean Water tongue of the northeast Atlantic we had to choose between SOFAR and RAFOS floats. The latter acronym, SOFAR spelled backwards, indicates the intimate relationship between the two systems. The inventor of this alternative, T. Rossby and his team (1986) state: "The technical difference is that whereas the SOFAR float transmits to moored receivers, the RAFOS float listens to moored sound sources".

The RAFOS system (Figs. 2.1 - 2.3) consists of bulky and expensive sound beacons (derived from former SOFAR floats), and handy and affordable underwater drifters (Zenk, 1990). An array of at least three moored sound sources is essential for an experiment (Fig. 2.4). Typically, mooring sites are several hundred kilometers apart. The sources transmit 80-second-long coded signals at pre-determined hours,typically, three times a day. A sound carrier frequency of 260 Hz with a modulation of 1.5 Hz is used. The acoustic signals are radiated omnidirectionally. They are received, decoded, and recorded by the drifting RAFOS floats. As in the case of the SOFAR floats the absolute time difference between transmission and reception allows the precise underwater navigation of the float. In addition to the arrival times from surrounding sound sources, floats record local pressure and temperatures data.

At the end of an oceanographic mission, a ballast weight is released from the up-to-then neutrally buoyant float. It returns to the sea surface where it starts transmitting its collected data set via satellite. Typically floats transmit every 45 s fractions of their memory content in a random loop fashion. After a few days the data redundancy allows a complete reconstruction of the original time series. The number of radio messages is limited only by the battery capacity, which lasts at least four weeks. Normally it takes only a few hours to obtain first float results that reach the institute in Kiel via a DATEX-P connection with the ARGOS host computer in Toulouse, France.

Usually RAFOS floats are built as expendable instruments. However, in one case a RAFOS float from Kiel was accidentally discovered and rescued near Madeira by a Portuguese coast guard ship. The instrument was identified by its internal label (Fig. 2.5) and was returned to Kiel.

Fig. 2.2a RAFOS float on board FS POSEIDON.

Fig. 2.2b RAFOS float in the tank laboratory of the Institut für Meereskunde in Kiel.

Fig. 2.3 Sound sources consist of two aluminum tubes. The long tube contains an electronics package and the batteries. The shorter upper pipe acts as a resonator for the transducer at its end. Sound sources used by IfM Kiel were bought from WRC, Falmouth, Mass., USA.

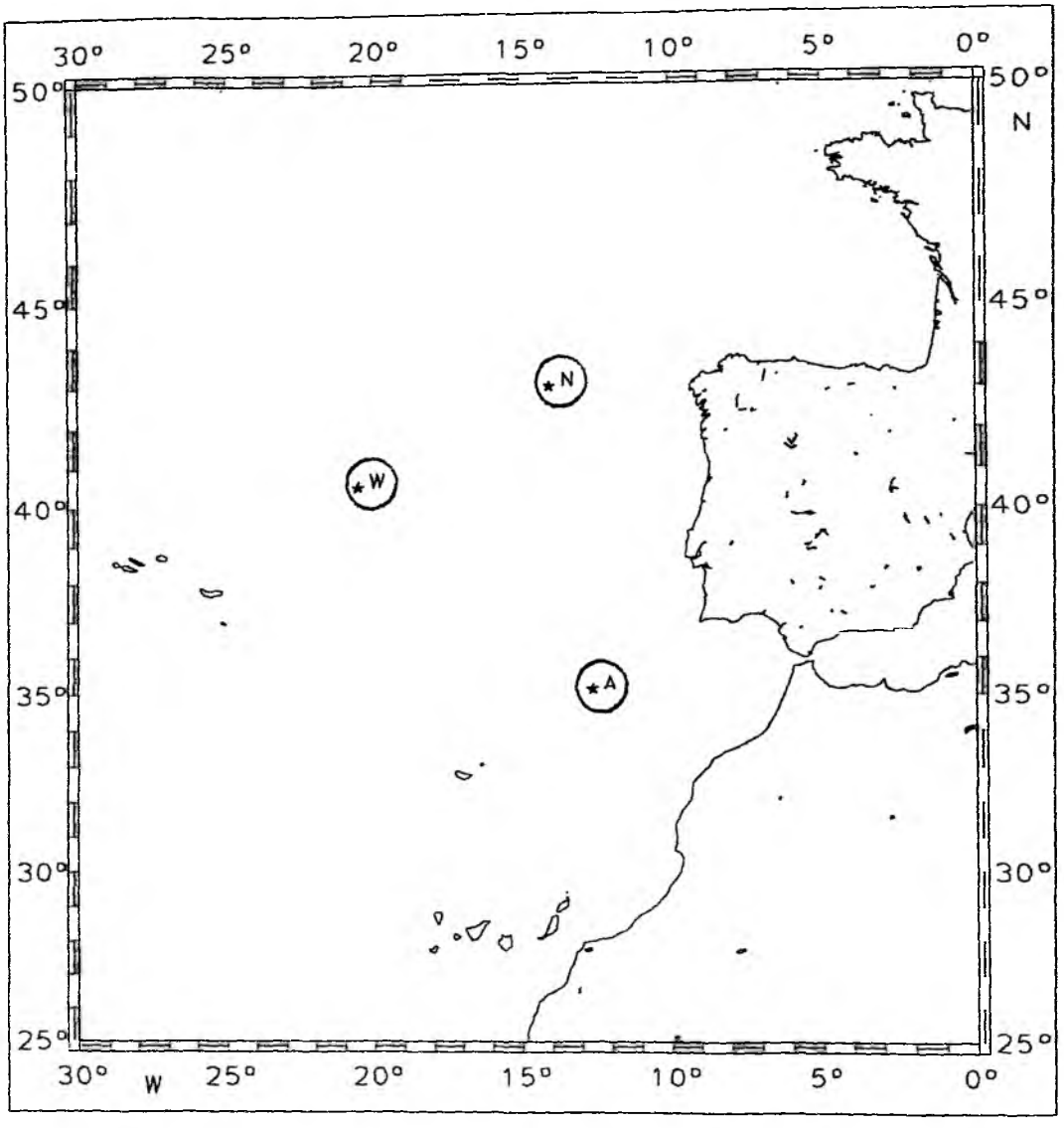


Fig. 2.4 The RAFOS technology presently (1991) is used by IfM Kiel in the Iberian Basin. It consists of an array of three sound sources. The western source is provided by IFREMER, Brest, France.

Lieber Finder !
Dies ist eine satellitengeortete Driftboje. Die Boje wurde zu Forschungszwecken ausgesetzt.
Bitte bergen Sie die Driftboje und informieren Sie uns über die ID-Nummer, den Fundort und den Zustand der Boje.
Wir sorgen für den Rücktransport, Ihre Auslagen werden Ihnen erstattet. Herzlichen Dank für Ihre Mühe.

Dear finder ,
This is a satellite tracked drifting buoy. The buoy was launched for research purposes.
Please recover the buoy and inform us about the ID number, the location of finding and the condition of the buoy.
We will arrange for its return and cover your expenses.
Thank you very much.

Estimado amigo !
Este instrumento que acaba de encontrar es una boya de deriva rastreada por satélite y utilizada en investigación.
Por favor, le agradecerîamos si recoge la boya y nos informa acerca del número de identificación (ID) , la localización y las condiciones en las que se encuentra.Nosotros organizaremos su devolución asi como las gastos que le haya originado. Muchas gracias .

Caro amigo !
Este instrumento que acabou de encontrar é uma boia de deriva rastreada por satélite e utilizada em pesguisa.
Por favor , nós o agradecerîamos se recolhesse esta boia e nos informasse o seu número de identificação (ID) , a localização em que a encontrou e as as condições em que ela se encontra. Nós organizaremos uma maneira de reave-la, assim como a devolução dos seus gastos. Muito obrigado .

Adress :

Institut für Meereskunde
an der Universität KIEL
Abt. Meeresphysik
Düsternbrooker Weg 20
D2300 KIEL 1
Fed. Rep. Germany

ID Nr. : 13

Tel. 0049-431-597-3891
Telex 17-431793
Fax 0049-431-565876

Fig. 2.5 RAFOS floats can easily be labelled by notes stored inside the glass housing. Multi-lingual messages are used in Kiel. In one case a float was accidently discovered and returned to Kiel.

3 RAFOS Float Modules

The configuration of a RAFOS float resembles an oversized test tube turned upside down. A schematic diagramm is shown in Figure 3. For details of the original RAFOS design from the University of Rhode Island, we refer to the informative article by Rossby et al. (1986).

3.1 Glass Pipe

The housing of a RAFOS float primarily consists of a DURAN (TM) glass pipe, as frequently used in chemical and food industries. The upper end of the pipe is shaped like a hemisphere while the lower end is ground flat. The tube is closed by a stainless steel endplate. Glass pipe and endplate are attached watertight to each other by a commercial silicone rubber. Several holes are drilled in the endplate, which are described later. Physical properties of DURAN (TM) glass are summarized in Table 3.1.

The choice of glass for underwater housings compared to steel or aluminum cages has several advantages:

* Glass is corrosion-resistant in saltwater,

* glass, especially borosilicate glass like DURAN (TM) with a low metal content, is transparent to electromagnetic radiation and magnetic fields,

* glass has a small thermal expansion coefficient,

* optical data links enable two-way communications without the need for opening any capped plugs, and

* readable captions can be easily attached inside the glass pipe (cf. Fig. 2.5).

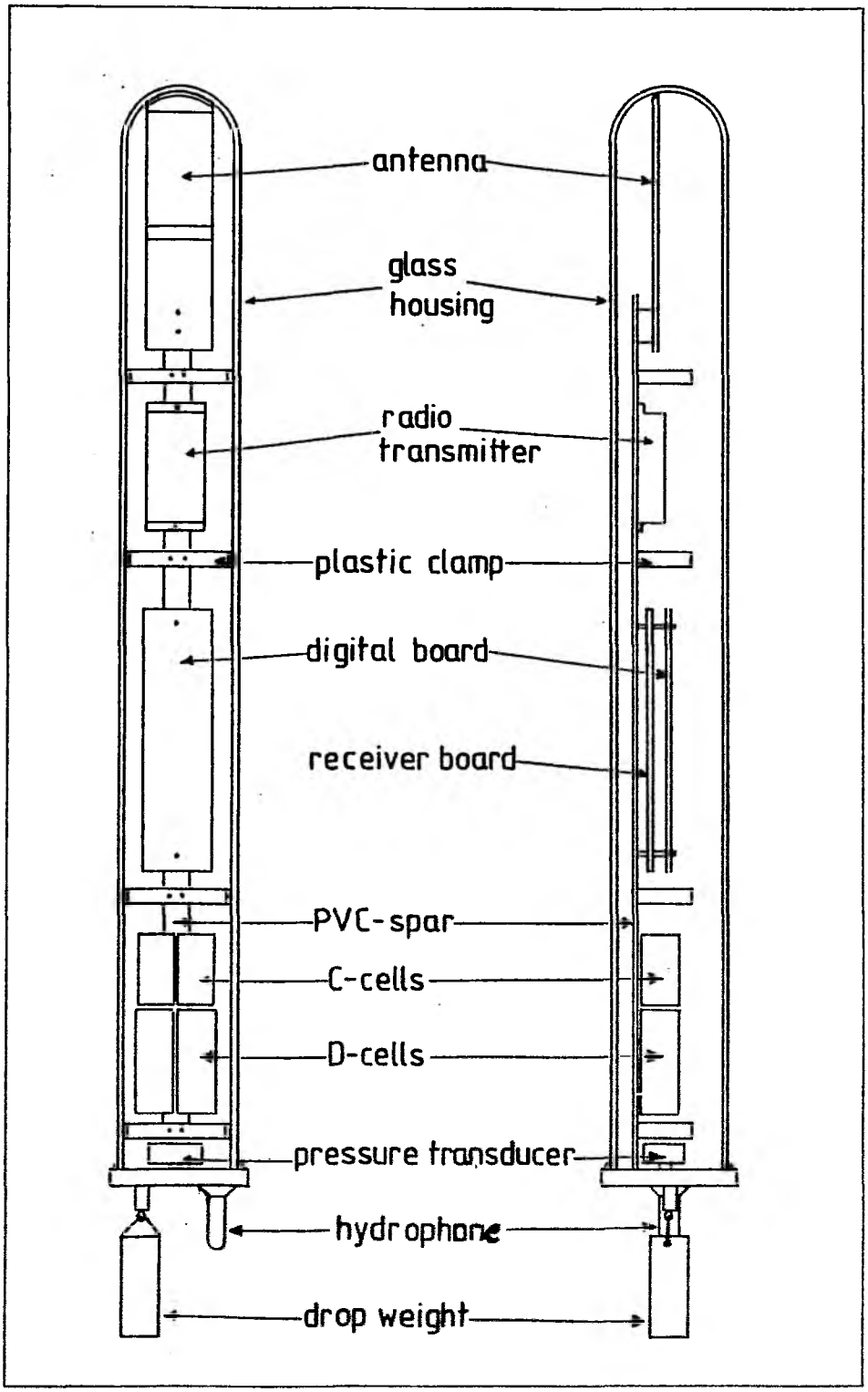


Fig 3.1 RAFOS float construction at IfM Kiel. The basic layout is identical with the original RAFOS float described by Rossby, Dorson, and Fontaine (1986).

(T)
DURAN

Physikalische Eigenschaften ***Physical properties***	**Wert** *Value*	**Einheit** *Unit*
Mittlerer lienarer Ausdehnungskoeffizient α 20/300 (nach DIN 52328) *Mean linear expansion coefficient*	3,25	10^{-6} K^{-1}
Transformationstemperatur *Transformation temperature*	530	°C
Viskositätstemperaturen *Viscosity temperature*		
$10^{13.0}$dPa s - obere Kühltemperatur - *Upper cooling temperature*	560	°C
$10^{7.6}$ dPa s - Erweichungstemperatur - *Softening temperature*	815	°C
10^{4} dPa s - Verarbeitungstemperatur - *Processing temperature*	1270	°C
Kurzzeitig höchstzulässige Gebrauchstemperaur *Short term maximum operation temperature*	500	°C
Dichte *Density*	2,23	g/cm³
Elastizitätsmodul E *Young's modulus*	63	10^{3} N/mm²
Poisson-Zahl μ *Poisson number*	0,20	1
Spezifische Wärmespannung φ *Specific heat tention*	0,26	nmm^{-2} K^{-1}
Wärmeleitfähigkeit λ (gültig für 90°C) *Heat conductivity*	1,16	W/m ·K
Temperatur für den spezifischen elektrischen Widerstand von $10^{8}\Omega$·cm(DIN 52326) t_{k100} *Temperature for the specific electrical resistance 10 $^{8}\Omega$·cm*	248	°C
Logarithmus des elektrischen Volumen-widerstandes bei 250°C	7,9	Ω·cm
bei 350°C	6,5	Ω·cm
Logarithm of the electrical volume resistance @250°C and @350°C		
Dielektrische Eigenschaften (1 MHz, 25°C) *Dielectric properties (1 MHz, 25°C)*		
Dielektrizitätszahl *Relative dielectric constant*	4,7	1
Dielektrischer Verlustfaktor tan δ *Dielectric residual loss*	55	10^{-4}
Brechzahl (λ=587,6nm) nd *Refraction index*	1,473	1
Spannungsoptische Konstante (DIN 52314)K *Optical tension constant*	4,0	10^{-6} mm²/N

Paul F. Schröder & Co. . Technische Glaswerke (GMBH & CO) . D-2086 Ellerau . Postfach 1153, Buchenweg 20 . Tel-SA-Nr.: (04106)7001-0 . FAX: (04106)700150 . Telex: 214888 SWRH

Table 3.1: Physical properties of DURAN (TM) glass, used for pressure houses of RAFOS floats (courtesy of Paul F. Schröder and Co., Ellerau).

The glass pipe for a given target depth requires the following characteristics (lenght, diameter, wall thickness, availability etc.):

* High mechanical steadiness under the target hydrostatic pressure,

* optimization of the ratio float weight to payload,

* at the end of mission the radio antenna of the float must be exposed sufficiently (~40 cm) above the water surface.

Considering all these factors we determined the optimum parameters for our present purposes to be 145 cm in length, outer diameter of 10 cm and a wall thickness of 0.5 cm (Gray and Stachiv, 1969; Young, 1989).

In May 1989 we performed several "o.k. tests" on board FS POSEIDON. Empty floats were pressurized in the deep sea using wire lengths of 1600, 1800, and 2000 m. Although statistically not significant we obtained evidence that our glass tubes can be used at a target depth of up to 1600 m. The theoretical burst pressure is of the same order. The 2000 m test was terminated by a powerful implosion.

3.2 Endplate and Sealing

As explained above the lower end of the glass pipe is closed by a stainless steel endplate. The first floats were equipped with aluminum endplates. But even short term missions of several days showed extremely strong corrosion problems, although the aluminum plates were hard anodyzed. Therefore we had to switch to stainless steel. Several tests showed that the attachment of the endplate to the glass pipe with a commercial silicone rubber (Sista F106) is sufficient. No O-rings are used. Another possibility would have been to construct a connection between endplate and glass pipe with shrink hose. However, attempts showed unacceptably high temperatures (>50°C) at the endplate, which could damage sensitive parts.

The endplate contains four drill holes and their respective threads for different modules:

* The hydrophone (Benthos AQ16) is used to receive coded signals from moored sound sources. The hydrophone mainly consists of a ceramic element which is cast in a two component resin.

* Another element on the endplate is the release plug. This is a metal bolt which is drilled through. In that bolt a special wire (INCONEL625) is potted with a two component resin. The wire is electrically insulated against the bolt and is shaped as a little loop at the lower end. Just prior to the float launch an individually calculated ballast weight is attached to that loop. At the end of the submerged mission 15V are applied to that wire. It then acts as an anode against the metal endplate (cathode). About 270 mA flow through the sea water, which results in an electrolytic dissolution of the INCONEL anode within 20 minutes. Thus the drop weight is released and the float returns to the surface.

* The third device on the endplate is the pressure transducer (Data Instruments, model EAF2K). It is a high gain strain gage device with a frequency-modulated square wave output with a nominal 5KHz range, set from 1 - 6 KHz. Other specifications are: hysteresis ±0.2% FS, long term stability ±1%/a, and rated overload pressure maximal 1.5x.

* Unlike the URI float there is a capped four-pin plug (Lemosa type RA) in the endplate. Three pins are used for the terminal connection. The fourth pin can be supplied with any float signal for external checking purposes.

In a more recent version we have introduced a two-way optical communication link, which may make the Lemosa connector obsolete in the near future. This way we save machining costs for the end plate and avoid unexpected leakage problems, potentially caused by the plug cap.

3.3 Arrangement of the Modules in the Float

The inner modules are mounted on a PVC spar (nickname: backbone). These parts, which are described in detail in the next section, are the battery pack, two electronics boards, the ARGOS transmitter and at the top the ARGOS antenna. The PVC spar extends nearly over the whole length of the float. Four plastic clamps on the spar press against the inner wall of the glass pipe. In this way the whole arrangement is protected against misalignment.

3.3.1 Battery Pack

For vertical stability the battery pack must be mounted at the lower end. It consists of ten 1.5 V D-cells and ten 1.5 V C-cells. These two groups of ten batteries are connected in series (2 x 15 V). In addition every float contains a separate AA-cell (1.5 V) for the internal clock. We use alkaline cells by VARTA (1979) with the following, manufacturer specified properties:

D-cell:	VARTA 4020 (Monozelle LR20)	1.5V;	@200hm and 20°C;	>8 Ah
C-cell:	VARTA 4014 (Babyzelle LR14)	1.5V;	@400hm and 20°C;	>4 Ah
AA-cell:	VARTA 4006 (Mignonzelle LR6)	1.5V;	@400hm and 20°C;	>1.5 Ah

The D-cells support the electronics and the sensors, whereas the C-cells are used at the end of the submerged mission. They are needed for releasing the ballast weight and for the radio transmitter. This guarantees enough power for surfacing and data transmission. Since there is no external switch for the battery pack, power is supplied by an internal Reed switch. By removing an external magnet off the Reed switch, which can be seen easily through the glass housing, the float is activated.

3.3.2 Electronics

The complete electronics are contained on two circuit boards commercially available from Bathy Systems, Inc. They are a receiver board (type 1203) and a digital board (type 1276) (Figure 3.2 and Tab.3.2). Pressure, temperature and sound signals are collected on the receiver board. Temperature is measured using a thermistor (Y.S.I., type 44032) soldered onto the receiver board. On the digital board, incoming data are processed and stored in RAMs (Random Access Memory). The heart of the whole float is the microprocessor MOTOROLA 6805E3E. It regulates, controls and monitors every event during the float mission. For energy saving reasons the electronics are only activated during programmed measurement cycles. Between measurement cycles the float enters into a sleep mode.

During their active mode RAFOS floats listen for accoustic signals emitted from moored sound sources. The sources transmit a frequency modulated signal with a carrier of about 260 Hz. The signal itself consists of a 80-s continous wave pulse, "centered near 260 Hz during which time the frequency increases linearly by 1.523 Hz" (Rossby et al., 1986). The received signal is demodulated twice (heterodyne receiver) before it can be digitised by the microprocessor. The applied method ensures a precise separation of the signal from the carrier with an optimal signal to noise ratio (Webb, 1977).

The electrical output from the hydrophone reaches the low noise amplifier ((1) on Fig. 3.2b) where its frequency ${}^{c}f$ is sent to a first intermediate mixer with a local frequency f_{z1} (3). Only the differential frequency

$$ {}^{i}f = f_{z1} - {}^{c}f \qquad (*) $$

can pass through the band pass filter (5, 6). ${}^{i}f$ reaches the second intermediate mixer with a local frquency f_{z2} (8). Again, only the differential frequency

$$ {}^{d}f = f_{z2} - {}^{i}f \qquad (**) $$

can pass through the band pass filter (7). The combination of (*) and (**) yields the relation between the modulated carrier (${}^{c}f$) and the demodulated signal for digitizing (${}^{d}f$).

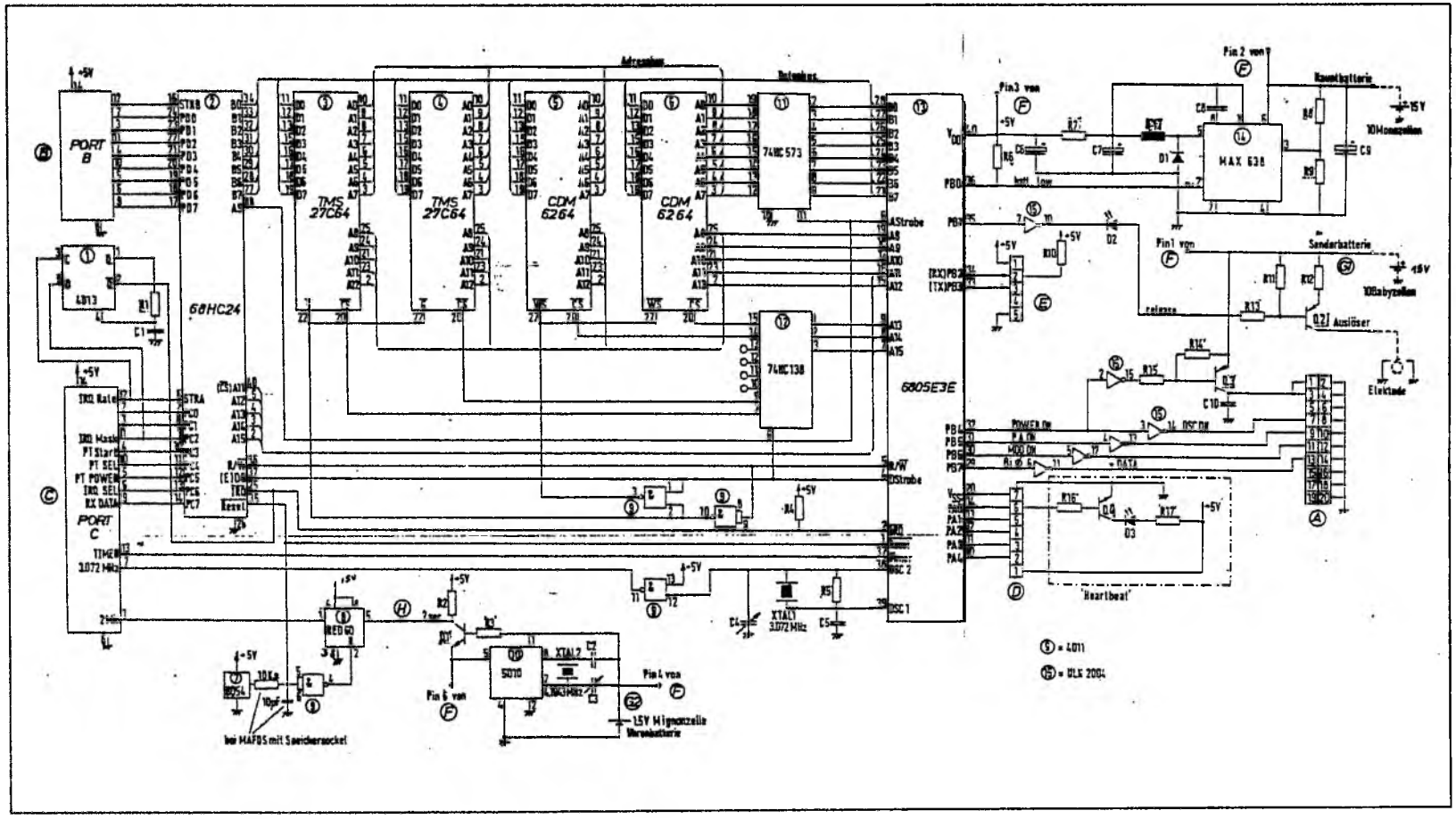


Fig. 3.2a Circuit diagram of the processor board (courtesy of Bathy Systems, Inc., West Kingston, R.I., USA).

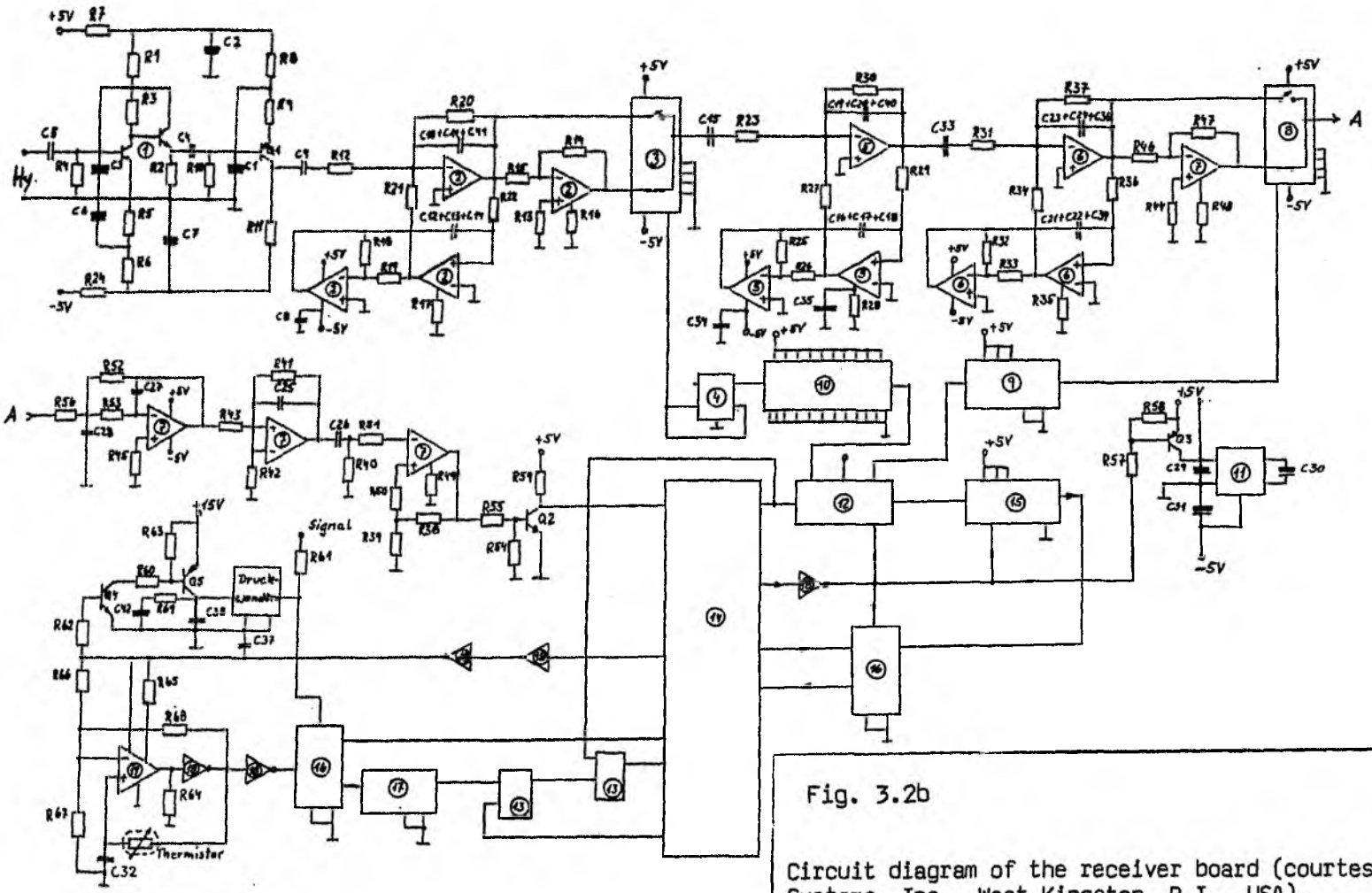


Fig. 3.2b

Circuit diagram of the receiver board (courtesy of Bathy Systems, Inc., West Kingston, R.I., USA).

Processor Board

Resistences

R 1 = 100 K Ohm
R 2 = 15 K Ohm
R 3 =270 K - 1 M Ohm
R 4 = 10 K Ohm
R 5 = 10 M Ohm
R 6 = 150 K Ohm
R 7 = 2.7 Ohm
R 8 = 1 M Ohm
R 9 = 100 K Ohm
R10 = 470 K Ohm
R11 = 100 Ohm
R12 = 2.7 Ohm
R13 = 680 Ohm
R14 = 1 K Ohm
R15 = 470 Ohm
R16 = 47 K Ohm
R17 = 1 K Ohm
R-101 = 9 x 100 K Ohm

Capacitors

C 1 = 220 p F
C 2 = 15 p F
C 3 = 2- 8 p F
C 4 = 15-60 p F
C 5 = 50 p F
C 6 = 100 F
C 7 = 100 F
C 8 = 0.1 n F
C 9 = 4.7 F
C10 = 0.1 n F

Coil

L1 = 1 m H

IC

1 =	CD 4013	D-FLIP-FLOP
2 =	68HC24	PORT
3 =	TMS 27C64	EPROM
4 =	TMS 27C64	EPROM
5 =	CDM 6264	RAM
6 =	CDM 6264	RAM
7 =	8054	
8 =	RED 60	Divider 60
9 =	CD 4011	NAND-Gatter
10 =	ICM7050IPA	Quartz-Clock
11 =	MC 74HC573	8 bit-Latch
12 =	MC 74HC138	Demultiplexer
13 =	6805E3E	Microprocessor
14 =	MAX 638	5V-Controller
15 =	ULN 2004	Inverter

Transistors

Q1 = 2N5089
Q2 = 2N5195 o. BD138
Q3 = 2N5193 o. BD138
Q4 = BC109

Diodes

D1 = 5817
D2 = LED
D3 = LED

Quartzes

XTAL1 = 3.072 MHz
XTAL2 = 4.1943 MHz

Misc.

Reedkontakt

Receiver Board

Resistences

R 1 = 49.9 K Ohm
R 2 = 162 K Ohm
R 3 = 162 K Ohm
R 4 = 221 K Ohm
R 5 = 2.21 K Ohm
R 6 = 332 K Ohm
R 7 = 1.21 K Ohm
R 8 = 332 K Ohm
R 9 = 1.21 K Ohm
R10 = 221 K Ohm
R11 = 221 K Ohm
R12 = 75 K Ohm
R13 = 49.9 K Ohm
R14 = 100 K Ohm
R15 = 100 K Ohm
R16 = 2.2 M Ohm
R17 = 2.2 M Ohm
R18 = 3.92 K Ohm
R19 = 3.92 K Ohm
R20 = 909 K Ohm
R21 = 26.1 K Ohm
R22 = 26.1 K Ohm
R23 = 75 K Ohm
R24 = 1.21 K Ohm
R25 = 3.92 K Ohm
R26 = 3.92 K Ohm
R27 = 39.2 K Ohm
R28 = 2.2 M Ohm
R29 = 39.2 K Ohm
R30 = 909 K Ohm
R31 = 75 K Ohm
R33 = 3.92 K Ohm
R34 = 39.2 K Ohm
R35 = 2.2 M Ohm
R36 = 39.2 K Ohm
R37 = 909 K Ohm
R38 = 100 K Ohm
R39 = 100 Ohm
R40 = 100 K Ohm
R41 = 100 K Ohm
R42 = 2.21 K Ohm
R43 = 56.2 K Ohm
R44 = 49.9 K Ohm
R45 = 681 K Ohm
R46 = 100 K Ohm
R47 = 100 K Ohm
R48 = 2.2 M Ohm
R49 = 2.2 M Ohm
R50 = 150 K Ohm
R51 = 56.2 K Ohm
R52 = 475 K Ohm
R53 = 475 K Ohm
R54 = 47 K Ohm
R55 = 47 K Ohm
R56 = 475 K Ohm
R57 = 47 K Ohm
R58 = 47 K Ohm
R59 = 15 K Ohm
R60 = 22 K Ohm
R61 = 10 K Ohm
R62 = 15 K Ohm
R63 = 10 K Ohm
R64 = 1 M Ohm
R65 = 49.9 K Ohm
R66 = 49.9 K Ohm
R67 = 49.9 K Ohm
R68 = 49.9 K Ohm
R69 = 100 Ohm

Table 3.2 Parts list for the RAFOS

Capacitors

C 1	=+	4.7	F	
C 2	=+	33	F	
C 3	=+	4.7	F	
C 4	=	2.7	nF	
C 5	=	0.1	F	
C 6	=+	4.7	F	
C 7	=+	33	F	
C 8	=	0.1	F	
C 9	=	0.1	F	
C10	=	22	nF	
C11	=	1	nF	(A)
C12	=	1	nF	(A)
C13	=	22	nF	
C14	=	750	pF	5% (A)
C15	=	1	F	
C16	=	5.6	nF	(A)
C17	=	100	nF	
C18	=		nF	5%
C18	=		nF	5%
C19	=	100	nF	
C20	=	5.6	nF	(A)
C21	=	100	nF	
C22	=	4.7	nF	(A)
C23	=	5.6	nF	(A)
C24	=	100	nF	
C25	=	220	nF	
C26	=	4.7	Fx	
C27	=	47	nF	
C28	=	220	nF	
C29	=+	33	F	
C30	=+	33	F	
C31	=+	33	F	
C32	=	6.2	nF	2%
C33	=	1	F	
C34	=	0.1	F	
C35	=	0.1	F	
C36	=	330	nF	5% (A)
C37	=	0.1	F	
C38	=	0.1	F	
C39	=	470	pF	5% (A)
C40	=	680	pF	5% (A)
C41	=	750	pF	5% (A)
C42	=	100	F	

IC

IC 1 = FH8736/LM394
IC 2 = OP 346CN
IC 3 = MC14053
IC 4 = MC14013
IC 5 = OP 346CN
IC 6 = -"-
IC 7 = -"-
IC 8 = MC14053
IC 9 = RED 360
IC10 = CD4059
IC11 = ICL7660
IC12 = MC14040
IC13 = MC14013
IC14 = PORT A
IC15 = RED 360
IC16 = MC14053
IC17 = MC14024
IC18 = MC14584
IC19 = OP CA3080

Transistors

Q1 = BC177
Q2 = BC107
Q3 = BC177
Q4 = BC107
Q5 = BC177

Thermistor

Y.S.I. 44032

electronics.

$$\boxed{{}^{d}f = f_{z2} - f_{z1} + {}^{c}f} \qquad (***)$$

The two local frequencies are constant and are given by the receiver hardware (Fig. 3.2b).

$$f_{z1} = 298.5 \text{ Hz}$$
$$f_{z2} = 40.0 \text{ Hz.}$$

With this information (***) can be written as

$${}^{d}f = ({}^{c}f/\text{Hz} - 258.5) \text{ Hz.}$$

Now consider the signal itself. Independant of the transmission and demodulation process the signal uses a band width Δf which is given by the difference between the upper and the lower edge frequency.

$$\Delta f = {}^{x}f_u - {}^{x}f_l \qquad (\S)$$

With a given set of band mid frequencies ${}^{x}f_m$ (x = (c, i, d)) we obtain

$$\begin{Bmatrix} {}^{x}f_u \\ {}^{x}f_l \end{Bmatrix} = \frac{\Delta f}{2} * \begin{Bmatrix} 1 \\ -1 \end{Bmatrix} + {}^{x}f_m$$

According to Rossby et al. (1986, 1992) Δf and ${}^{c}f_m$ are constant,

$$\Delta f = 1.523 \text{ Hz,}$$
$${}^{c}f_m = 260.1365 \text{ Hz.}$$

We obtain

$$\begin{Bmatrix} {}^{x}f_u \\ {}^{x}f_l \end{Bmatrix} = ({}^{x}f_m/\text{Hz} + \begin{Bmatrix} 0.7615 \\ -0.7615 \end{Bmatrix})\text{Hz} \qquad (\S\S\S)$$

An evaluation of (§§§) is shown in Table 3.3

f/Hz rounded	^{x}f		
	^{c}f	^{i}f	^{d}f
f_u	260.898	39.125	2.399
f_m	260.137	38.364	1.637
f_l	259.375	37.602	0.876

Table 3.3 Frequency map of the RAFOS heterodyne receiver circuit in Fig. 3.2b. Indices u, m, and l denote the upper edge, the mid and the lower edge of the band pass cascades. The three indices x represent the carrier, the intermediate and the twice demodulated frequency. In all cascade steps the bandwidth $\overline{\Delta}f = {}^{x}f_u - {}^{x}f_l$ remains constant.

Before the signal with ^{d}f has reached the line driver (Q_2) it has been hard limited by a comparator (7-right). This step excludes amplitude variations from the signal. Via port A the signal is digitized at a 10 Hz rate by the microprocessor. Since sound sources transmit 80 s-long CW signals the digitizing process delivers 800 bit. Fig. 3.2c-f depict various time dependant aspects of these frequencies and signals. Fig. 3.2c represents the continuous frequency increase during the active 80 s transmission $^{c}f(t)$, i.e. the linear sweep. $^{i}f(t)$ and $^{d}f(t)$ represent the intermediate frequency and the frequency of the demodulated signal:

$$\boxed{{}^{d}f(t) = \left(\frac{1.523}{80s} \cdot t + 0.876\right) \text{ Hz}}$$

Since the analog signal $^{d}f(t)$ is sampled at 10 Hz, in the following we deal with 800 discrete values of step width 0.1 s:

$$0.1 < t < 80.$$

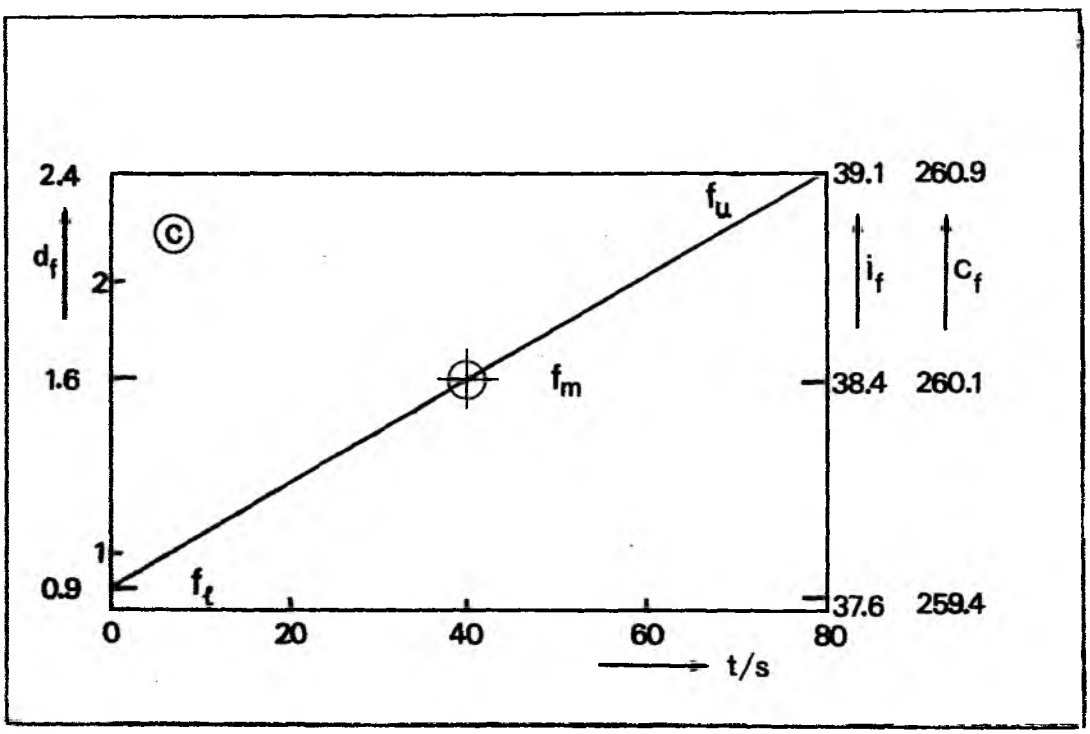


Fig. 3.2c During their transmission sound sources change the carrier frequency ^{c}f between $^{c}f_l$ and $^{c}f_u$. In two steps (via ^{i}f to ^{d}f) we obtain the demodulated frequency ^{d}f. f_m represents the mid frequencies.

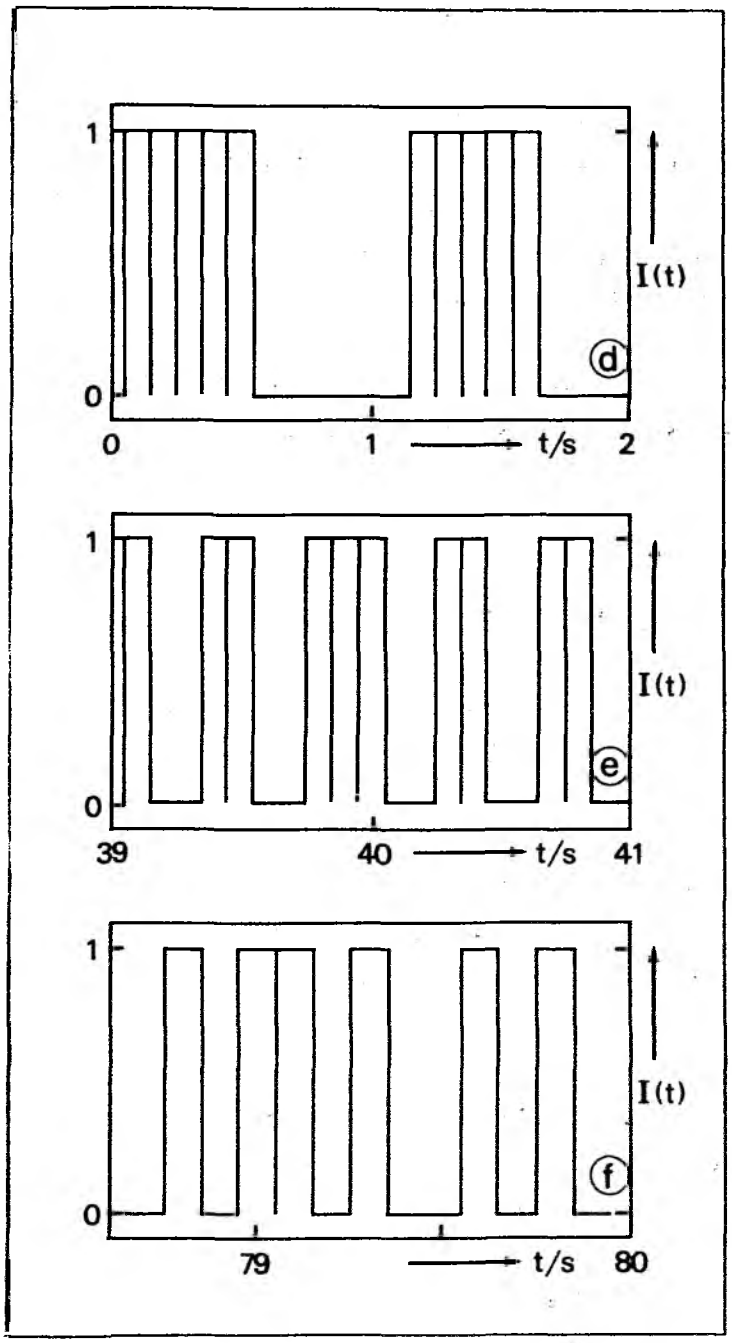


Fig. 3.2-d-f Three snapshots of the digitised continuous wave signal I from the beginning (d), the middle (e), and the end (f) of the 80s long sound transmission.

The hard limited signal (varying between Hi and Low only) can be simulated by

$$I(t) = 1/2\ (1 + \text{sign}\ (\sin\ (2\pi\ {}^{d}f(t)t))).$$

Since not all 800 values of I(t) could be resolved in one graph with sufficient resolution, in Fig. 3.2d-f we show examples of I-distributions from the beginning, from the middle part, and from the ending signal I(t). Each column in the graphs represents one bit, just as the microprocessor would see an error free data train I(t).

Correlation of the incoming, i.e. real signal R(t) with the ideal signal I(t) is achieved in the microprocessor by an XOR function:

R	0	1	0	1
I	0	0	1	1
XOR (R, I)	0	1	1	0

Finally the sum S of all XOR operations is calculated:

$$S = \sum_{1}^{800} \text{XOR}\ (R,\ I).$$

If we sampled a 80s-CW-signal with 10 Hz a perfect signal (identical to the reference) we would obtain $S_p = 0$. In contrast, random input data would give the no-signal value $S_r = 400$.

For a given listening window, the real time [Time 1] at which the expression abs($S - S_r$) reaches a maximum together with its value [Cor 1] are stored in the RAM. In addition the second highest value [Cor 2] is recorded with the corresponding time [Time 2].

Format descriptions follow in Table 5.3, where [Cor #] is called "correlation height" and [Time #] "arrival time". Corr 2 and Time 2 allow a recording of a second sound source signal during one listening window (cf. section 5.3). In order to discriminate both signals Cor 1 and Cor2 in one window the corresponding sound sources should be several hundred kilometers apart.

3.3.3 ARGOS Transmitter

The ARGOS transmitter is manufactured by Toyo Communication Equipment Co. (type T-2023). Since formating for data transmission is an integral part of the float firmware (see below), no separate controller between a data storage device and the transmitter is necessary. After activating the transmitter approximately every 45 s one 32-byte-message (ARGOS message) is transmitted. The repetition rate of about 45 s is derived from an individual identification code. This platform ID number is provided by Systeme ARGOS for all data telemetering platforms. The fact that the repetition rate is a function of the ID number ensures that two floats never have identical repetion rates for their transmissions. The first two bytes of an ARGOS message contain a checksum and a sequence number. The remaining 30 bytes are reserved for measured data. Once the whole memory is transmitted the transmissions start again from the beginning. This cycle is repeated until the batteries are depleted. More details on data formats are given in Sections 5.6 and 5.7.

We use a cheap printed circuit board as an antenna. Since the transmission carrier frequency lies at 401 MHz a length of around 40 cm is sufficient for the antenna ($\lambda/2$ dipol).

3.3.4 Float Firmware

FORTH is used as the programming language for RAFOS floats (Carter, 1986). This highlevel computer language (cf. Kelly and Spiess, 1986) was chosen for two main reasons:

* FORTH is more flexible than ASSEMBLER. Necessary adjustments to the float mission can be made immediately before launching the float.

* For programming, any dumb terminal is sufficient. An ASSEMBLER code on a separate downloading computer is unnecessary.

On the digital board we find two EPROMs (Eraseable Programmable Read Only Memory, TI TMS 27C64) and two RAMs (RCA CDM 6264) as shown in Fig. 3.3.

All four chips have 8 kbyte = 8192 bytes of memory each. For a more detailed description of the memory map we refer to Figure 3.3.4 (Carter, 1989). EPROM #1 contains the relative addresses from $E000 to $FFFF, EPROM#2 $C000 to $DFFF. Hexadecimal numbers are labeled by "$". Some memory adjustments must be made before the final assembly of the EPROMs on the circuit board (cf. Section 4.1).

RAM A contains the address space $0000 to $1FFF. $0000 to $01D9 is occupied by miscellaneous FORTH. $01D9 is the beginning of the dictionary space when the system is powered up and it extends to $0218. During a mission FORTH reuses this space as scratch space when accepting input, so the values at those locations always change. In addition to the dictionary space, there is a 'pad' area. The pad area extends to address $03B2. The following 60 bytes up to $03EE are called scratch pad. Including a safety distance of 93 bytes the FORTH code needs space from address $0000 to address $044B. The 7091 bytes from address $044C to $1FFF are free memory. This means about 86% of the total memory space of RAM A is available and can be used for data storage.

RAM B covers the adresses $2000 to $3FFF. Addresses from $2000 to $27FF (= 2048 bytes) are reserved for the main data storage buffer. For the data storage format see section 5.6. It is important to know that this storage buffer is open-ended. That means there is no explicit check whether it reaches the end of the allocated data storage space! The VOlume CHAnger data buffer (presently not used in Kiel) begins at address $2800. From that address to $3D1B (= 5404 bytes), the data for the vocha cycle is stored if the vocha parameter VPTR is set to 1 (default is 0; for VPTR see Appendix 4: FORTH glossary). Like the data storage buffer, the vdata buffer is open-ended. When programming a mission it is very important to stop the mission before the data reaches address $3C80! The RAM space from $3C80 to $3FFF is used by the FORTH code. The default mission needs the space from address $3C80 to $3D7F for some variables and the acoustic receiver signal. The RAM space from $3D80 to $3FFF contains the TERMINAL INPUT BUFFER (TIB), the RETURN STACK and the PARAMETER STACK.

EPROM #1	EPROM #2	RAM B	RAM A
E000	C000	2000	0000
E1D2		free memory	} miscellaneous Forth
} rom code	} mission B	(used by RAFOS	01D9
F5C7		and MAFOS)	} dictionary space
F5CE			0219
}mission A			
FBEF			} pad
FC0B	CCB2	} data buffer	03B2
} mag			} scratch pad
FD7F			03EE
FD80		2800	} safety distance
} cor irq			044C
FE2D		} vadta buffer	
FE2E			free memory
} look table		3C80	
FF2D	D000	} LTT	(used exclusively
FF2E		3CF8	by MAFOS)
} reference	} mission B	} signal data	
FF91		3D7F	
FF92	D021	3D80	
} Q reference		} tib	
FFF5		3DFF	
FFF6 w time		3E00	
FFF8 time		} return stack	
FFFA ext int		3EFF	
FFFC warm int vector		3FF0	
FFFE cold int vector		} stack	
FFFF	DFFF	3FFF	1FFF

Fig. 3.3 Extract from the memory map of RAFOS floats. All addresses are given in hexadecimal code (courtesy of S. Carter, Electra Software Consulting, Pacific Grove, CA, USA).

If the vocha is passive, the VDATA buffer can be used together with the main data storage buffer. So there are 7452 bytes (= 90 % of total RAM B space) for data storage. The practical mission length is explained in Section 5.4.

4. Float Construction in Kiel

4.1 Assembly

Since the electronic boards are delivered without any firm- or software, they have to be equipped with two EPROMs, containing the FORTH software. This RAFOS FORTH code was originally developed at URI (Carter, 1986). Before final assembly some changes have to be made in both EPROMs. As before all hexadecimal numbers are labeled by "$".

* First, each float has an individual identification number (ID), which is supplied by the French company CLS (ARGOS), Toulouse. This ID number is located in EPROM #2, at addresses $091B, $069F and $06A0. As an example Fig. 4.1a shows the ID number for float $55CD8. The ID number consists of five characters. It is 20 bits long. The top four bits ($5) are located at address $091B. The high nibble there is $F. This should never be changed. It is nescessary to comply with ARGOS requirements. The final 16 bits are at $069F ($5C) and at $06A0 ($D8) located. When communicating with the float via terminal it only uses the lower 16 bits as its ID. But it actually transmits all five digits to the ARGOS satellite (and also to the Telonics test receiver, see Section 4.2 for Telonics format).

* The next modification has to be made in EPROM #1. At address $1942 and $1943, the warm-up time is stored of the pressure transducer, see Fig. 4.1b. In the original EPROM from URI this time was set to 20 deciseconds ($14). But it turned out that this time appeared to be too short for the pressure transducer to deliver a stable frequency output. For Kiel floats this value is set to 100 deciseconds ($64).

Adresse	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
$00890	EC	61	EE	FD	FF	EE	C8	3D	E2	04	03	4D	53	47	20	20	■aЄ² Є╚═Γ--MSG
$008A0	20	C8	63	BC	80	C7	5B	E6	36	00	07	E8	70	E5	39	E6	╚c╝C├┤u6--♦pσ9u
$008B0	36	00	1E	E7	CB	EE	D0	EA	6A	C7	B6	E4	2A	E8	70	C7	6--τ╥╞╩╝j╟╢╪Σ*♦p╟
$008C0	9A	E4	2A	E8	70	E4	01	C7	5B	E6	36	00	08	E8	70	EA	ÜΣ*♦pΣ-├┤u6--♦pΩ
$008D0	6A	E6	70	E5	39	EE	FD	FF	E0	C8	6C	C7	5B	E6	36	00	j♦pσ9Є² α╚╢├┤u6-
$008E0	06	E8	70	E5	39	E6	36	00	1E	C7	B6	E5	D6	E2	04	09	-♦pσ9u6--╢╪σ╥Γ--
$008F0	52	41	44	5F	53	54	C8	9A	BC	80	F3	87	E6	36	C6	36	RAD ST╚Ü╝C¢cu6╠6
$00900	E5	B3	E6	36	00	11	E5	0B	E6	36	00	14	E5	0B	F3	7B	σ│u6--σ-u6--σ-≤{
$00910	E6	36	FF	FE	C7	5B	E5	0B	E6	36	2F	F5	C7	5B	E8	17	u6 ▪├┤σ-u6/⌡├┤♦-
$00920	E5	0B	C6	9C	C7	5B	E6	36	00	04	E8	70	E5	0B	E7	CB	σ-╞£╟├┤u6--♦pσ-τ╤
$00930	C7	B6	E5	0B	F9	CE	C7	9A	E5	0B	C7	E5	E2	04	02	4E	╟╢σ-·╪├┤Üσ-├┤σΓ--Ñ
$00940	4F	20	20	20	20	C8	EF	CC	ED	67	00	00	03	55	53	45	Ö ╚╦╠╘φg---USE
$00950	20	20	20	C9	3E	CC	ED	67	00	02	07	46	4C	41	53	48	╒>╠╘φg---FLASH
$00960	45	C9	4C	BC	80	E7	CB	E5	39	E2	04	04	53	45	45	44	E╒L╝Cτ╤σ9Γ--SEED
$00970	20	20	C9	5A	CC	ED	67	3D	16	06	52	41	4E	44	4F	4D	╒Z╠╘φg=--RANDOM
$00980	C9	6B	BC	80	C9	74	E4	2A	E6	36	7A	B7	E6	36	7F	FF	╒k╝C╒tΣ*u6z╖u6-

Adresse	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	E	0123456789ABCDEF
$00650	08	9A	BC	9E	06	4F	53	43	5F	4F	4E	C6	3F	BC	80	E6	-Ü╝R-08C ON╞?╝Cu
$00660	36	00	1B	E7	E3	E5	39	E2	04	07	43	41	52	52	5F	4F	6--τπσ9Γ--CARR O
$00670	C6	54	BC	80	E6	36	00	3D	E7	E3	E5	39	E2	04	06	54	╞╢╝Cu6-=τπσ9Γ--T
$00680	52	5F	4F	46	46	C6	69	BC	80	E6	36	00	0D	E7	E3	E5	R OFF╞i╝Cu6--τπσ
$00690	39	E2	04	02	49	44	20	20	20	20	C6	7E	CC	ED	67	5C	9Γ--ID ╞~╠╘φg\
$006A0	D8	07	49	44	5F	48	41	53	C6	93	BC	80	C6	9C	EB	03	╪-ID HAS╞ôC╝╞£╦♥-
$006B0	E6	36	7A	B7	E6	36	7F	FF	F1	1F	E4	9E	E6	16	00	A0	u6z╖u6- ±-ΣRu6-á
$006C0	F1	5A	E6	36	01	C2	E8	70	E2	04	05	44	45	4C	41	59	±Zu6-┬♦pΓ--DELAY
$006D0	20	C6	A1	BC	80	E7	CB	F8	13	C6	AA	E7	CB	EE	D0	F3	╞í╝Cτ╤°-╞¬τ╤╩╪╩╬
$006E0	93	EE	FD	FF	FC	F7	F2	E2	04	07	57	41	52	4D	5F	55	ôЄ² ⁿ≈ΣΓ--WARM Ü
$006F0	C6	CA	BC	80	E6	36	04	00	E7	CB	EE	D0	F3	93	EE	FD	╞╩╝Cu6--τ╤╩╪╩╬ôЄ²
$00700	FF	FC	E2	04	06	57	31	36	30	4D	53	C6	E9	BC	80	E6	ⁿΓ--W160MS╞Θ╝Cu
$00710	36	00	80	E7	CB	EE	D0	F3	93	EE	FD	FF	FC	E2	04	05	6-Cτ╤╩╪╩╬ôЄ² ⁿΓ--
$00720	5A	45	52	4F	3F	20	C7	04	BC	80	E5	B3	EB	46	E1	DB	ZERO? ╟-╝Cσ│δFß╛
$00730	00	08	E3	F0	E6	36	00	FF	E2	04	03	31	50	21	20	20	--π≡u6- Γ--1P!
$00740	20	C7	1F	BC	80	E5	B3	E4	2A	E4	9E	E6	82	E5	0B	E2	╟-╝Cσ│Σ*ΣRu♠σ-Γ

Fig. 4.1a Location of memory specifics. Location of the platform ID number in EPROM #2. The example shows the position of ID $55CD8. It is 20 bits long (cf. Table 5.4). The top four bits of the ID are located at $091B as the low nibble. The high nibble is $F = const. The additional 16 bits are located at $069F and at $06A0.

Adresse	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
$018E0	FE	E2	04	08	53	59	4E	43	46	4C	F8	CA	CC	ED	67	00	▪Γ--SYNCFL°≡╣Θg-
$018F0	0F	05	50	53	59	4E	43	20	F8	E3	BC	80	E7	CB	F8	EC	--PSYNC °π╝C╤╥°m
$01900	E5	39	F3	7B	F3	93	E5	B3	F8	EC	E4	01	EB	8A	E1	DB	σ9≤{≤δσ|°mΣ-6è8▄
$01910	FF	F4	F3	87	E3	F0	E2	04	04	53	59	4E	43	20	20	F8	⌠≤c╤≡Γ--SYNC °
$01920	F1	BC	80	F8	EC	E4	01	E1	DB	FF	FA	F3	87	E2	04	E2	±╝C°mΣ-8▄ -≤cΓ-Γ
$01930	04	06	53	41	4D	50	4C	45	F9	18	BC	80	F8	13	F7	A5	--SAMPLE--╝C°-≈╗
$01940	E6	36	00	14	F8	FA	F7	F2	F6	A4	F8	D3	F7	92	E6	36	µ6--°-≈≥÷ñ°╜≈Γµ6
$01950	01	00	E6	36	00	08	E4	01	E8	9A	E6	36	00	43	E5	39	--µ6--Σ-╣Θµ6-Cσ9
$01960	E6	36	00	42	E4	2A	E2	04	03	54	4F	44	20	20	20	F9	µ6-BΣ*Γ--TOD ▪
$01970	31	CC	ED	67	00	27	05	54	49	4D	45	3F	20	F9	68	BC	1╠Θg-'-TIME? ▪h╝
$01980	80	E6	36	00	27	E4	2A	F1	BF	E6	36	00	3C	F1	32	E6	Cµ6-'Σ*±¿µ6-<±2µ
$01990	36	00	64	ED	B0	E8	70	EC	B2	E2	04	03	4C	54	50	20	6-dΘ╝╣pm▓Γ--LTP
$019A0	20	20	F9	76	CC	ED	67	00	0E	04	54	43	4F	52	20	20	▪v╠Θg---TCOR
$019B0	F9	9B	CC	ED	67	00	0C	04	52	53	55	4D	20	20	F9	A9	▪¢╠Θg---RSUM ▪┌
$019C0	CC	ED	67	00	43	04	44	41	54	41	20	20	F9	B7	CC	ED	╠Θg-C-DATA ▪┐╠Θ
$019D0	67	04	4D	05	56	44	41	54	41	20	F9	C5	CC	ED	67	28	g-M-VDATA ▪┼╠Θg(

Adresse	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
$018E0	FE	E2	04	08	53	59	4E	43	46	4C	F8	CA	CC	ED	67	00	▪Γ--SYNCFL°≡╣Θg-
$018F0	0F	05	50	53	59	4E	43	20	F8	E3	BC	80	E7	CB	F8	EC	--PSYNC °π╝C╤╥°m
$01900	E5	39	F3	7B	F3	93	E5	B3	F8	EC	E4	01	EB	8A	E1	DB	σ9≤{≤δσ|°mΣ-6è8▄
$01910	FF	F4	F3	87	E3	F0	E2	04	04	53	59	4E	43	20	20	F8	⌠≤c╤≡Γ--SYNC °
$01920	F1	BC	80	F8	EC	E4	01	E1	DB	FF	FA	F3	87	E2	04	E2	±╝C°mΣ-8▄ -≤cΓ-Γ
$01930	04	06	53	41	4D	50	4C	45	F9	18	BC	80	F8	13	F7	A5	--SAMPLE--╝C°-≈╗
$01940	E6	36	00	64	F8	FA	F7	F2	F6	A4	F8	D3	F7	92	E6	36	µ6-d°-≈≥÷ñ°╜≈Γµ6
$01950	01	00	E6	36	00	08	E4	01	E8	9A	E6	36	00	43	E5	39	--µ6--Σ-╣Θµ6-Cσ9
$01960	E6	36	00	42	E4	2A	E2	04	03	54	4F	44	20	20	20	F9	µ6-BΣ*Γ--TOD ▪
$01970	31	CC	ED	67	00	27	05	54	49	4D	45	3F	20	F9	68	BC	1╠Θg-'-TIME? ▪h╝
$01980	80	E6	36	00	27	E4	2A	F1	BF	E6	36	00	3C	F1	32	E6	Cµ6-'Σ*±¿µ6-<±2µ
$01990	36	00	64	ED	B0	E8	70	EC	B2	E2	04	03	4C	54	50	20	6-dΘ╝╣pm▓Γ--LTP
$019A0	20	20	F9	76	CC	ED	67	00	0E	04	54	43	4F	52	20	20	▪v╠Θg---TCOR
$019B0	F9	9B	CC	ED	67	00	0C	04	52	53	55	4D	20	20	F9	A9	▪¢╠Θg---RSUM ▪┌
$019C0	CC	ED	67	00	43	04	44	41	54	41	20	20	F9	B7	CC	ED	╠Θg-C-DATA ▪┐╠Θ
$019D0	67	04	4D	05	56	44	41	54	41	20	F9	C5	CC	ED	67	28	g-M-VDATA ▪┼╠Θg(

Fig. 4.1b The warm-up time for the pressure transducer is located in EPROM #1 at $1942 and $1943. Originally this time was set to 20 ds (= $14, top). In the Kiel floats it was adjusted to 100 ds (= $64, bottom).

Adresse	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
$01950	01	00	E6	36	00	08	E4	01	E8	9A	E6	36	00	43	E5	39	--µ6--Σ-♦Ûµ6-Cσ9
$01960	E6	36	00	42	E4	2A	E2	04	03	54	4F	44	20	20	20	F9	µ6-BΣ*Γ--TOD ·
$01970	31	CC	ED	67	00	27	05	54	49	4D	45	3F	20	F9	68	BC	1╠íg-'-TIME? ·h¾
$01980	80	E6	36	00	27	E4	2A	F1	BF	E6	36	00	3C	F1	32	E6	Cµ6-'Σ*±¿µ6-<±2µ
$D1990	36	00	64	ED	B0	E8	70	EC	B2	E2	04	03	4C	54	50	20	6-dí═♦p═╣Γ--LTP
$019A0	20	20	F9	76	CC	ED	67	00	0E	04	54	43	4F	52	20	20	·v╠íg---TCOR
$019B0	F9	9B	CC	ED	67	00	0C	04	52	53	55	4D	20	20	F9	A9	·¢╠íg---RSUM ·⌐
$019C0	CC	ED	67	00	43	04	44	41	54	41	20	20	F9	B7	CC	ED	╠íg-C-DATA ·╖╠í
$019D0	67	20	00	05	56	44	41	54	41	20	F9	C5	CC	ED	67	28	g --¥DATA ·┼╠íg(
$019E0	00	04	31	43	4F	52	20	20	F9	D3	CC	ED	67	3D	1A	05	--1COR ·└╠íg=--
$019F0	31	54	49	4D	45	20	F9	E1	CC	ED	67	3D	18	04	32	43	1TIME ·ß╠íg=--2C
$01A00	4F	52	20	20	F9	EF	CC	ED	67	3D	16	05	32	54	49	4D	OR ·∩╠íg=--2TIM
$01A10	45	20	F9	FD	CC	ED	67	3D	14	06	42	49	47	43	4F	52	E ·²╠íg=--BIGCOR
$01A20	FA	0B	CC	ED	67	3D	10	07	42	49	47	54	49	4D	FA	19	--╠íg=--BIGTIM--
$01A30	CC	ED	67	3D	12	05	56	46	4C	41	47	20	FA	27	CC	ED	╠íg=--¥FLAG -'╠í
$01A40	67	3D	0E	06	42	4F	54	54	4F	4D	FA	35	CC	ED	67	3D	g=--BOTTOM-5╠íg=

Adresse	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
$01950	01	00	E6	36	00	08	E4	01	E8	9A	E6	36	00	43	E5	39	--µ6--Σ-♦Ûµ6-Cσ9
$01960	E6	36	00	42	E4	2A	E2	04	03	54	4F	44	20	20	20	F9	µ6-BΣ*Γ--TOD ·
$01970	31	CC	ED	67	00	27	05	54	49	4D	45	3F	20	F9	68	BC	1╠íg-'-TIME? ·h¾
$01980	80	E6	36	00	27	E4	2A	F1	BF	E6	36	00	3C	F1	32	E6	Cµ6-'Σ*±¿µ6-<±2µ
$01990	36	00	64	ED	B0	E8	70	EC	B2	E2	04	03	4C	54	50	20	6-dí═♦p═╣Γ--LTP
$019A0	20	20	F9	76	CC	ED	67	00	0E	04	54	43	4F	52	20	20	·v╠íg---TCOR
$019B0	F9	9B	CC	ED	67	00	0C	04	52	53	55	4D	20	20	F9	A9	·¢╠íg---RSUM ·⌐
$019C0	CC	ED	67	00	43	04	44	41	54	41	20	20	F9	B7	CC	ED	╠íg-C-DATA ·╖╠í
$019D0	67	04	4C	05	56	44	41	54	41	20	F9	C5	CC	ED	67	28	g-L-¥DATA ·┼╠íg(
$019E0	00	04	31	43	4F	52	20	20	F9	D3	CC	ED	67	3D	1A	05	--1COR ·└╠íg=--
$019F0	31	54	49	4D	45	20	F9	E1	CC	ED	67	3D	18	04	32	43	1TIME ·ß╠íg=--2C
$01A00	4F	52	20	20	F9	EF	CC	ED	67	3D	16	05	32	54	49	4D	OR ·∩╠íg=--2TIM
$01A10	45	20	F9	FD	CC	ED	67	3D	14	06	42	49	47	43	4F	52	E ·²╠íg=--BIGCOR
$01A20	FA	0B	CC	ED	67	3D	10	07	42	49	47	54	49	4D	FA	19	--╠íg=--BIGTIM--
$01A30	CC	ED	67	3D	12	05	56	46	4C	41	47	20	FA	27	CC	ED	╠íg=--¥FLAG -'╠í
$01A40	67	3D	0E	06	42	4F	54	54	4F	4D	FA	35	CC	ED	67	3D	g=--BOTTOM-5╠íg=

Fig. 4.1c In the case of RAFOS floats the default start address for data storage is $2000. This constant (used in RAM B) is stored in EPROM #1 at $19D1 and $19D2 (top). For MAFOS missions this constant may be changed into $044C to be stored at the same location in EPROM #1 (bottom). In this case data storage starts already in RAM A.

* For the MAFOS application, a third change has to be made in EPROM #1 as well. During RAFOS float operations the data storage starts in RAM B at address $2000. Since for MAFOS operations (see Section 6.1) free memory space in RAM A can be for storage (see Fig. 3.3) data also can be stored in RAM A from address $044C. The variable for the start address of data storage is located at $19D1 and $19D2 in ERPROM #1 (Fig. 4.1c).

4.2 Control Instruments

After the float is assembled and sealed, it is connected to a terminal (or a personal computer with terminal emulation) via its serial port. Therefore an interface box is necessary, which converts the float signal level (0/+5V) to RS232 standards (+12/-12V). A terminal emulation software (Persoft, Smartern 240 (TM)) allows all communication parameters to be properly matched (1200 baud, 8 bit, no parity, 1 stop bit). This software also includes a capture option to log all terminal in- and outputs in a capture file. The collection of capture files represents an important documentation of all calibration, checking and starting activities.

Once the magnet is removed and the battery power is connected to the electronics, a red light emitting diode (called release LED) is turned on. After any key is hit the LED turns off. This LED is an optical verification of the current supplied to the release plug (see Section 5.5, LON and LOFF and also OFFSET).

Near the microprocessor a second, but smaller red LED is located (see Fig. 3.2). This is the "heart beat" LED. As explained in Section 5.1, every control step in the float is triggered by a positive transition of a 2-minute strobe. The heartbeat LED indicates this transition every 2 minutes by a short blink. Once a float is ready for launching, this LED is the one and only indicator that the float is still "alive". However, during a navigation window (see section 5.3.1) this LED is deactivated.

To check the ARGOS transmitter, an appropriate receiver is necessary. The IfM float group uses a Telonics ARGOS uplink receiver, Model TSUR-B. This receiver provides a RS232 output to log the received ARGOS messages on

computer disk. An example of the Telonics display data format is shown in Fig. 4.2 (cf. Fig. 5.9.1). The display format is derived from the ARGOS specified data format, reproduced in section 5.7.

```
01-30-90  15:33:02  ID  05494   DA
FF  FE  2F  F5  5D  94  B7  00
00  12  24  00  DD  00  00  21
27  01  08  18  00  CA  20  01
1D  16  00  1C  09  6A  02  22
20  00  F9  17  00  95
CT 182  DB  032 PO 001

01-30-90  15:33:47  ID  05494   DA
FF  FE  2F  F5  5D  94  3F  01
1B  01  3F  19  00  38  1B  00
6D  17  00  1A  07  6A  06  40
1D  01  66  1C  00  0A  1D  00
9C  19  00  22  1F  01
CT 182  DB 032  PO 000
```

Fig. 4.2: Two ARGOS messages sent by RAFOS float $55D94. Data were recorded by a Teleconics receiver. For details on format refer to text and the Telonics manual.

In the first line date, time, float ID (decimal!) and receiver operation mode are shown. Obviously the repetition rate of the float was 45 seconds. The first three bytes (0-2) in the second line are fixed in the RAFOS float. Bytes 3, 4 and 5 contain the float ID as hexadecimal numbers (refer to RAD_START in FORTH glossary). The conversion of the ARGOS ID number from the hexadecimal to the decimal system is achieved by the formula

$$\frac{\$\langle ID\rangle}{\$40} \Longrightarrow \langle ID\rangle$$

Byte number six ($B7) of the ARGOS message is the check sum calculated by the microprocessor from the remaining 31 bytes. All ARGOS messages are labeled in a consecutive order by the float. This message number is given by byte 7 ($00). The remaining 30 bytes in the third and the following lines are reserved for recorded data. All ARGOS/Telonics messages are organized in the same way. For details on the internal measurement data refer to Sections 5.6 and 5.7.

4.3 Calibration Work

For best results the pressure transducer, the thermistor, and the float oscillator have to be calibrated individually for every RAFOS float: temperature and pressure data originally are given as frequencies, which are measured by the time port (pin 37) of the microprocessor (cf. Fig. 3.2). The resulting digital raw data (counts) are then converted into standard units, i.e. in centibars and centidegree, respectively. The usage of these uncommon units is caused by the inability of FORTH (cf. 3.3.4) to handle floating point numbers. Values in centibars and centidegrees are both integer numbers.

The following formulas for intermediate pressure and temperature calibrations apply to all floats (in BASIC convention):

$$p\ [c\ bar] = (13560000 \backslash C) - 2756$$

$$temp\ [c\ degree] = (((((C * -921 \backslash 10000) + 880) * C) \backslash 1000 - 3550) * C) \backslash 1000 + 5850$$

C represent "counts" from the counter circuit.
\ stands for integer division with truncation.

Since the algorithms for both parameters are universal for all floats, the resulting data in standard units can only represent an estimate of the true value (Fig. 4.3). The final correction is done in Kiel after the estimated pressure and temperature data are received by Systeme ARGOS (cf. Section 8).

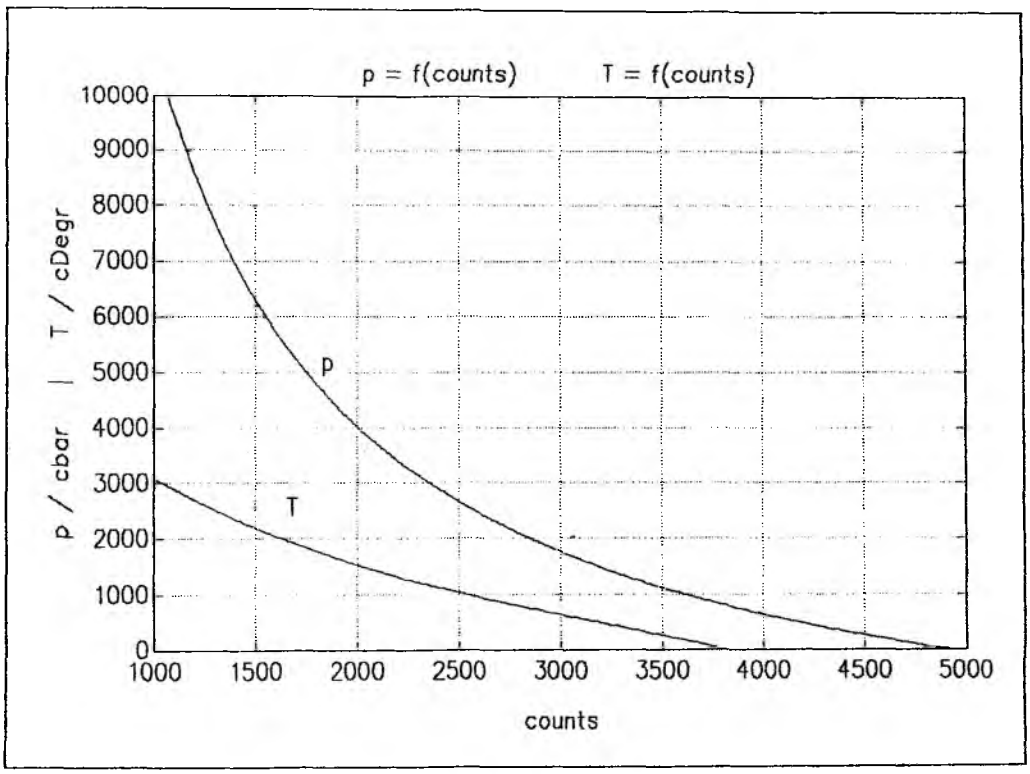


Fig. 4.3 Pressure and temperature calibration curves used for all floats. During data processing the resulting values in [c bar] and [c Degree C] are individually corrected by float specific calibration coefficients.

4.3.1 Pressure Calibration

For calibration of the pressure transducer with its subsequent electronics, the float is treated as a black box. A dead weight tester is connected to the pressure transducer. We advise mounting the pressure transducer on the end plate prior to calibration. We found that any mechanical stress on this sensor (including assembly) can affect its calibration curve. Pressure is increased from 100 dbar to 1300 dbar (pressure range of transducer EAF2K is restricted to 2000 psi = 1378 dbars) in 100 dbar steps. With the FORTH word - PP (refer to FORTH glossary) an averaged pressure is calculated out of ten FORTH controlled measurements for each pressure step. With these ten elementary measurements and the corresponding dead weight tester pressures, a linear regression is performed. Slope and y-intercept are stored for each float individually in its data file REDIT.DAT (see Section 8 for subsequent data processing). After a float has surfaced and transmitted its data these two calibration coefficients are applied to the raw pressure data transmitted from the float. We estimated a resulting accuracy of O(2) dbar.

4.3.2 Temperature Calibration

RAFOS floats are equipped with "Uni Curve" thermistors by Y.S.I. Within a range of 100 mK the manufacturer garantees interchangeability of the type 44032. In Table 4.1 we reproduce the manufacturer's specifications for this thermistor.

T/°C	0	5	10	15	29	25	30
R/KΩ	94.98	74.44	58.75	46.67	37.30	30.00	24.27

Table 4.1: Selected temperature vs resistance values for Y.S.I. thermistor 44032. The minimal dissipation constant of the used thermistor amounts to 1 µW/mK in still air. The dissipation constant is defined as the electrical power required to raise the thermistor temperature above the surrounding temperature (self-heating).

Initially the calibration of the thermistor circuit was achieved indirectly. One thermistor as part of the receiver board was calibrated in a temperature bath. With the resulting resistance vs temperature relation a resistor calibration box was built. It was used as a thermistor replacement device for the following temperature calibrations. Analogous to pressure calibrations, a linear regression was performed on ten elementary temperature measurements. However, from recent float data we have learned that this bulk temperature calibration (resistor box method) is insufficient for our purposes. Since then we introduced individual calibrations of each thermistor including its matching electronics. A universal resistor calibration box no longer is in use. The resulting present accuracy now is at least O(50) mK.

In a laboratory test we have determined temperature time constants of MAFOS monitors (König et al., 1991) to be ~90 min. A time constant is the time required for a thermistor to indicate 63% of a newly impressed temperature. This result of ~90 min is expected to be identical with the time constants of RAFOS floats. In both cases the thermistor is mounted on the receiver board without any direct contact to the end plate or the glass wall. These intentionally long time constants match the slow sampling rates of both instruments, which typically are of O(4-24 h).

4.3.3 Oscillator Calibration

The float clock (i. e. its oscillator) is slightly temperature dependent. The oscillator on the digital board is adjusted by the manufacturer at room temperature. Hence, it is necessary to tune the oscillator to the target temperature at which the float is expected to be launched. This is achieved by measuring the two second strobe with a counter at jumper H on the digital board and by adjusting capacitor C3 (see Fig. 3.2). The necessity for oscillator calibration is demonstrated in four different examples in Fig. 4.4, where temperature dependent clock errors within 2 s before and after adjustment of C3 are displayed. Since the time axis is given in 10^{-5} s we conclude that clock drifts of O(1) s/day may be expected, if proper adjustment were neglected.

Unfortunately during the submerged mission the float clock still may have a remaining drift of O(1) s/month. For the subsequent data processing this clock drift needs to be determined and applied to the calculation of the arrival times of sound signals. This clock drift can be determined by comparing the (programmed) float time and the exact time, which is added automatically to all ARGOS messages.

Let us suppose a float has finished its underwater mission. It has 40 ARGOS messages in memory and was programmed for data transmission to start at 5:00 (without any clock drift). A first message was received at 8:00 by satellite. Assuming that the float's repetition rate was determined to 50 seconds using a stop watch, it would have sent 216 messages during that three hours (10800 seconds). But this would be only correct if the repetition rate would be 50.000 seconds. If the repetition rate were 50.1 seconds a tremendous shift in time would happen. To transmit 216 messages with a repetition rate of 50.1 seconds would last 10821.6 seconds. Hence, the float would have transmitted for three hours and 21.6 seconds. A repetition rate of 50.01 seconds would result in a drift of time of 2.16 seconds.

The determination of the repetition rate to at least 10^{-3} will insure that the start of the transmission is calculated correctly. With a repetition rate of 50.001 seconds it takes 10800.216 seconds to transmit 216 messages. The repetition rate can be measured by a counter at pin 32 (signal OSC ON) of the microprocessor (see Fig. 3.2).

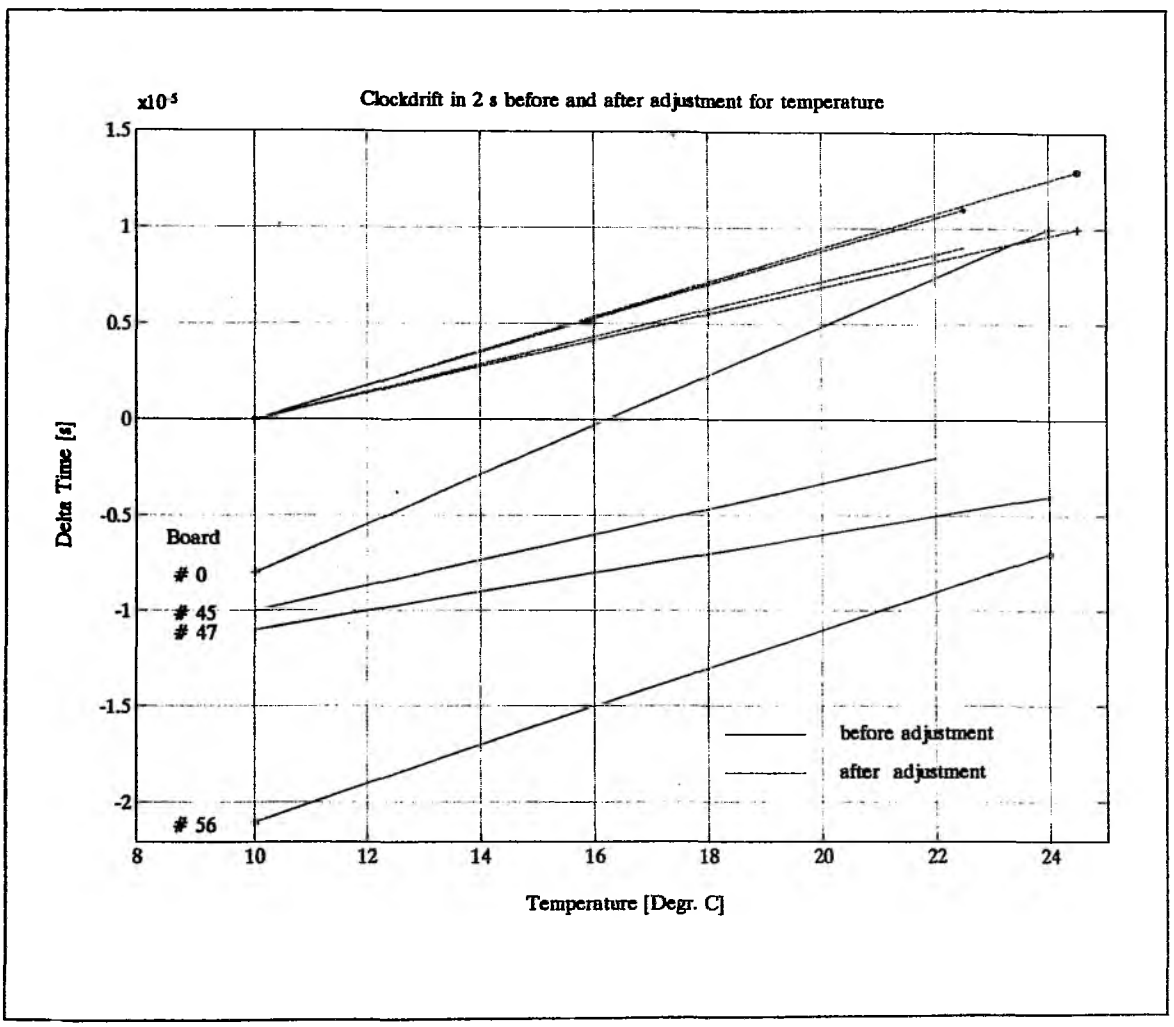


Fig. 4.4 Four examples of temperature dependent clock drift adjustments.

4.4 Ballasting

The principles of ballasting neutrally bouyant floats have been described by Swallow (1955). Our ballasting procedure was originally developed at the Graduate School of Oceanography at the University of Rhode Island. The principles of ballasting are explained in detail in Rossby et. al (1985).

The calculation of the additional drop mass Δm, to bring the float from the surface to its target depth is

$$\boxed{\Delta m = \rho_{is} \cdot V_{is} - \rho_{bt} \cdot V_{bt}} \quad (*)$$

where Δm = the additional mass to bring the float to its target depth

V_{is} = volume of float system at target depth,

V_{bt} = volume of float system in ballast tank,

ρ_{is} = density of water at target depth,

ρ_{bt} = density of water in ballast tank.

The subscripts refer to in situ and to ballast tank locations. The float volume at its target depth is a function of temperature and pressure:

$$\boxed{V_{is} = V_{bt} \cdot \left(1 + \frac{1}{V_{bt}} \cdot \frac{\partial V_{bt}}{\partial T} \cdot \Delta T + \frac{1}{V_{bt}} \cdot \frac{\partial V_{bt}}{\partial p} \cdot \Delta p\right)} \quad (**)$$

Using the definitions for the thermal expansion coefficient α and the compressibility coefficient Kappa

$$\alpha = \frac{1}{V} \frac{\partial V}{\partial T}\bigg|_p \; ; \qquad \text{kappa} = \frac{1}{V} \frac{\partial V}{\partial p}\bigg|_T$$

and by inserting (**) in (*) we obtain:

$$\boxed{\Delta m = V_{bt} \left[(\rho_{is} - \rho_{bt}) + \alpha \cdot \rho_{is} \cdot \Delta T + \text{kappa} \cdot \rho_{is} \cdot \Delta p\right]} \quad (***)$$

where

ΔT = temperature difference between target depth and ballast tank,
Δp = pressure difference between target depth and ballast tank.

V_{bt} is here called the volume of the "float system", because that volume V_{bt} includes the glass pipe, the drop weight and any other material which will be with the float during its mission.

The principle of float ballasting is shown in Fig. 4.5. Determining the additional drop weight by equation (***), we have to calculate V_{bt}. This is simply done by using the Archimedian principle.

The float is brought to a neutral bouyant state just a few centimeters above the tank bottom with a big drop weight, two ballasting chains, and, for fine trimming, a few washers. Two "ballasting chains" have been attached at the endplate with tape. Their weight per length is known. These two chains have two purposes:

* They serve as an anchor. The float can not drift out of the view of the observation camera (see below) and because two chains are used, the float can not turn around its axis and the paper scale inside the float will not turn out of the camera view.

* Later, when pressure is applied, the tank water becomes denser than the float, and the float will start to ascend. But with the two chains, the ascent is never out of control, because every moment when the float gets higher it takes more chain and therefore more ballast weight with it. In this way, the equilibrium point (neutral buoyancy) is always well defined under any pressure.

After the float is placed in the tank, the ballast tank is closed and pressure is applied. Two adjusted TV cameras (Philips VK4902) monitor the ascend ot the float system with increasing pressure. The length of the ballasting chain (as a function of the tank pressure) is recorded (Fig. 4.6). The ascent of the float is measured with the scale, which is fixed inside the transparent glass float. The upper camera image shows how many centimeters the float has risen and therefore how much chain the float has taken from the tank bottom. The lower camera allows an inspection of

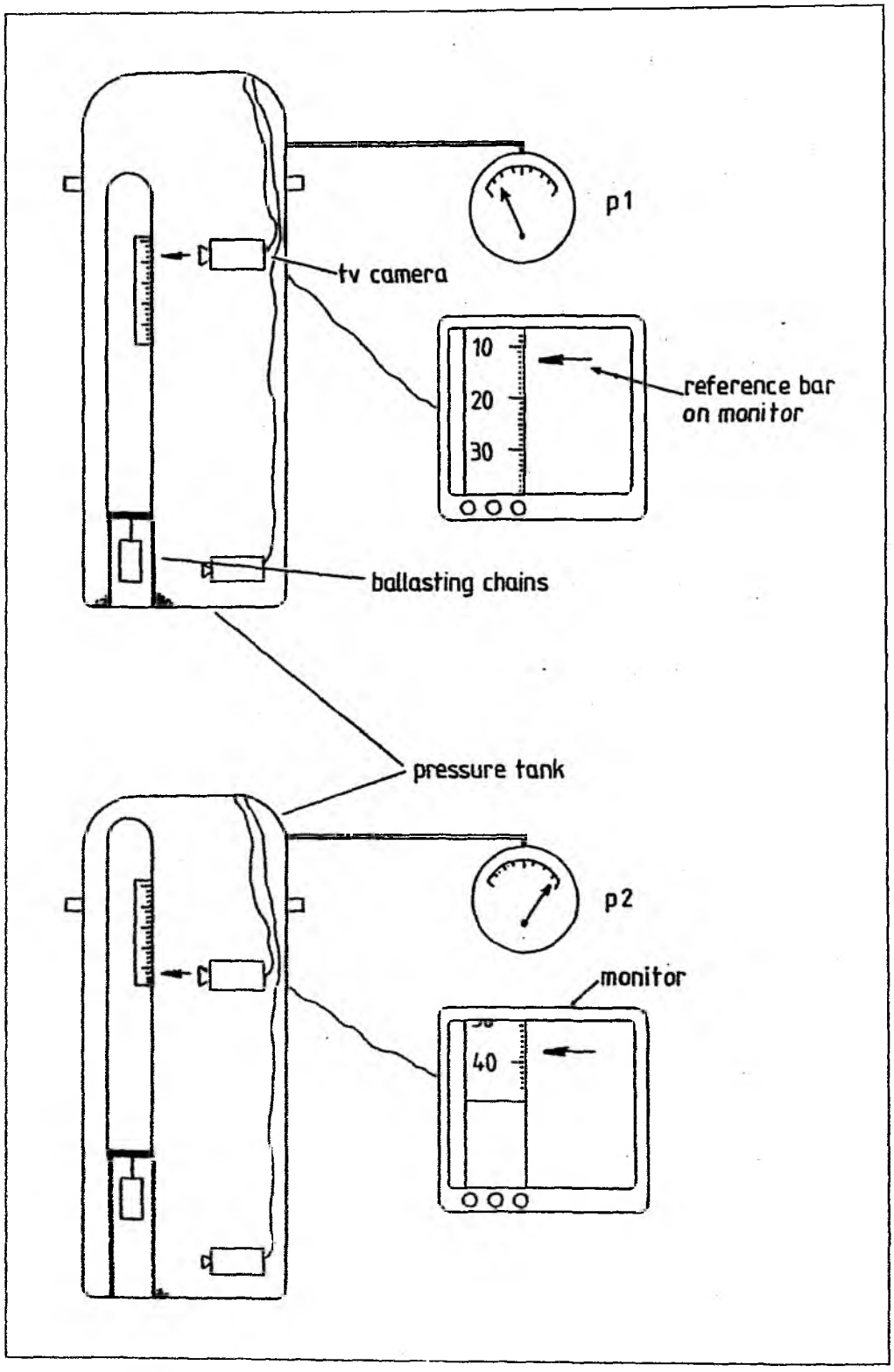


Fig. 4.5a Ballasting procedure and pressure tank facilities at IfM Kiel.
The maximum pressure in the test tank amounts to 40 bars.

Fig. 4.5b The pressure vessel contains an inner tube with two TV cameras. The top camera is visible on the right side. In the center the top of a submerged RAFOS float can be seen.

Fig. 4.5c Pressure tank control stand at IfM Kiel.

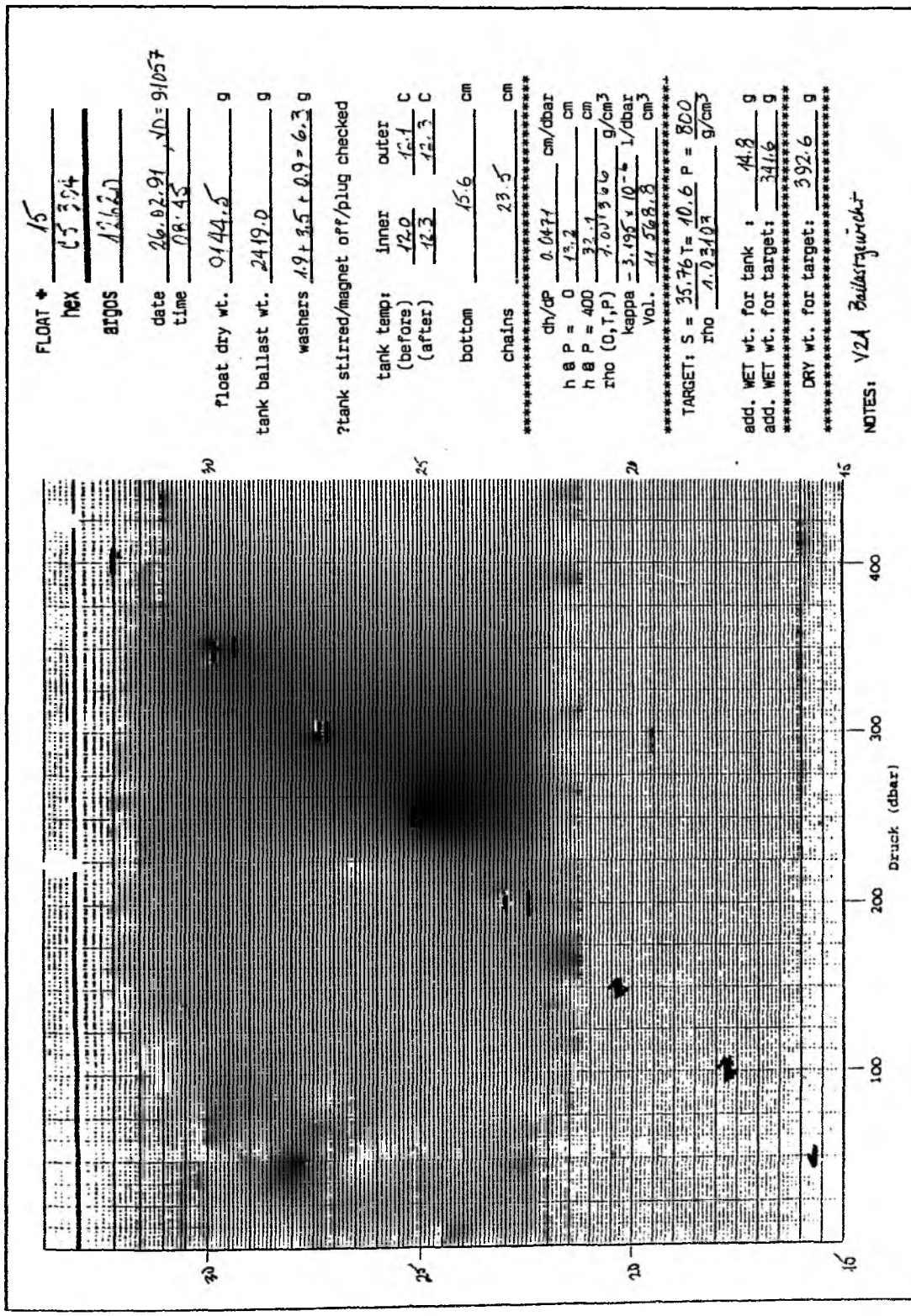


Fig. 4.6 Ballasting form developed for use at IfM Kiel. On the right side float data and ballasting procedures are recorded. On the left side a graph of the float height vs tank pressure is displayed.

the behavior of the anchor chains at the bottom. Pictures from both cameras can be monitored sequentially on one screen in the tank lab. The maximum pressure in the used test tank in Kiel amounts 40 bars.

Occasionally the float descends in the very first moment of pressurizing. This is due to the possibility that little air bubbles are adhered to the float, that not all mating surfaces and O-rings are pressed into place and that the silicon rubber is not pressed thoroughly to the glass pipe and the endplate, so that the float volume is not well defined. But with increasing pressure these points of uncertainty are removed and the float behaves as expected.

A linear regression is applied to the chain length vs tank pressure values. The chain length at (for example) atmospheric pressure (p = 0) can be converted into mass, so that the float volume is simply:

$$V_{bt} = \frac{m_{float} + m_{dropweight} + m_{washers} + m_{chains}}{\rho_{bt}} .$$

The density of the tankwater ρ_{bt} is calculated with the equation of state (Unesco, 1983). Initially the salinity of the tap water was assumed zero.

After the float volume is determined, the compressibility coefficient can be calculated with equation (***). For any tank pressure the additional mass Δm and the salt water density can be calculated using the equation of state. Often the temperature difference ΔT is small, so it is no significant error to put $\Delta T = 0$. This results in a simplified equation:

$$\boxed{\text{kappa} = \frac{\rho_{bt} - \rho_{is}}{\rho_{is}} + \frac{(\Delta m / V_{bt})}{\Delta p}}$$

Since our glass pipes are hand made, and the wall thickness of the glass is not constant from pipe to pipe, it is absolutely necessary to ballast each float individually. One example for a typical range of kappa is given in Fig. 4.6. The mean compressibility coefficient was found to be

$$\overline{\text{kappa}} = (-3{,}2587 \cdot 10^{-6} \pm 0{,}0566 \cdot 10^{-6})\ {}^{1}/\text{dbar}.$$

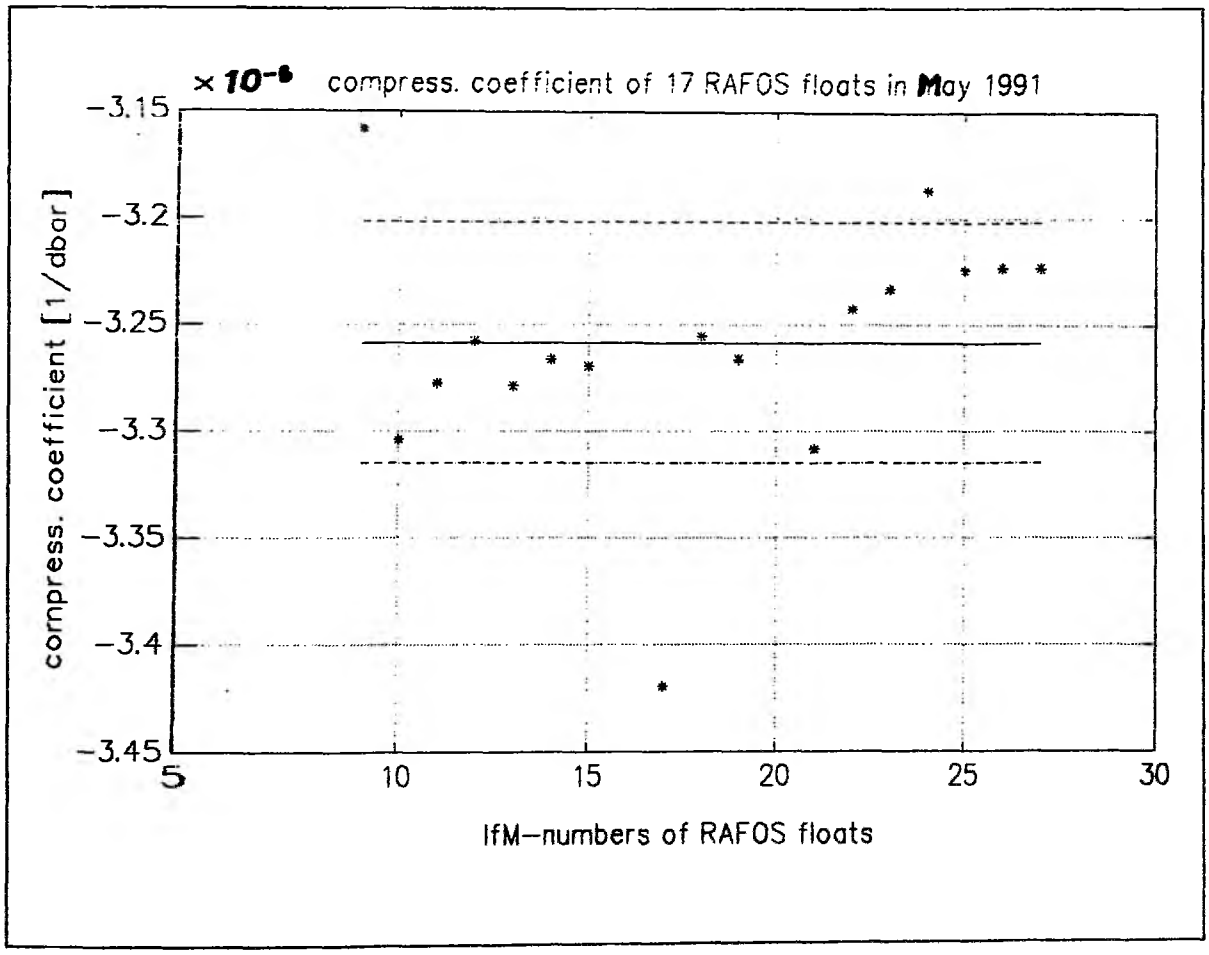


Fig. 4.7 Example for compressibilities of RAFOS floats. The solid bar indicates the mean. Dashed lines stand for standard deviation. Data were measured with the pressure tank shown in Fig. 4.5.

Our ballast tank in Kiel has no temperature control facility. Hence, it is not easy to measure the thermal expansion coefficient α under the given circumstances. It was measured only once ($\alpha = 6{,}44 \cdot 10^{-5}\ 1/K$). This result deviates substantially from α for pure DURAN (TM) glass according to Table 3.1. How much of this difference is caused by the other materials of the "float system" and how much by our determination method remains unclear. However, we conclude that this may be a critical item to be investigated further, especially when the temperature difference between the ballasting tank and the target depth exceeds several Kelvins.

Just prior to launching the float a temperature and salinity measurement should be taken at the target depth. The additional mass Δm for the target depth is calculated and could be made available prior to launch. The ballast chains do not go with the float on the submerged mission. Their mass is added to the final ballast weight.

In the laboratory we usually measure forces rather than masses; i.e. we calculate masses instead of measuring them. The additional mass for the drop weight Δm in equation (***), that accounts for the density difference between the tank water and the target depth causes different weight forces either in water (G_{is}) or in air (G_{air}). They are sometimes called "wet" and "dry" weight. Their relationship again can be derived from the Archimedean principle.

$$\boxed{G_{air} = G_{is} \cdot \frac{\rho_{dw}}{\rho_{dw} - \rho_{is}}}$$

with ρ_{dw} = density of the drop weight. To avoid corrosion (weight) losses we use stainless steel (V2A). We have determined its density repeatedly to be $\rho_{dw} = 7.9\ g\ cm^{-3}$, which agrees with standard text books.

5. Operations

As before it is esssential to distinguish between decimal and hexadecimal numbers. It is common to label hexadecimal numbers with "$", for example: 255 = $FF. In parts where communication with the float is discussed, all monitor outputs of the float are underlined.

5.1 Starting Procedure

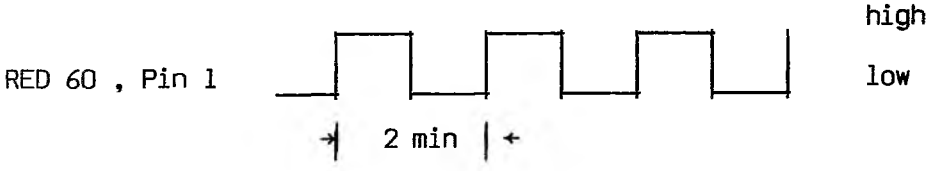
Since both sound sources and floats need exact time bases, an accurate real time clock is needed as a reference. To synchronize the float time base with the reference clock it is important to turn on the float power (remove the magnet) on an odd minute. The float itself has an internal 2-minute strobe, which is sometimes called "heartbeat". Every two minutes the float clock is triggered. This means that after starting on an odd minute the float clock has to be set to an even time! Example (all commands are terminated by the "return" key):

Real time clock	17:15:00	--> turn on float power.
Set the float clock	: "1714 Z"	
Query the time	: "TIME?"	
The float prompts	: "1714"	

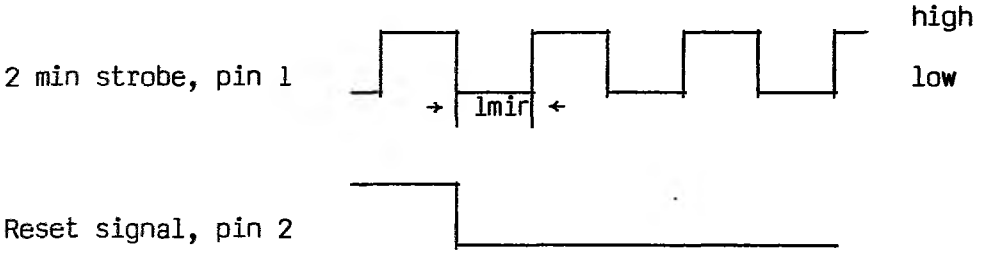
Every two minutes the float clock increases its time increment. For example at 17:27:55 real time the float clock is 17:26. At 17:28:00 the next heart beat occurs and the float clock is set to 17:28. Since the float has no internal real time clock, one can set the time in the float to any time which is necessary for any purposes. But it has to be set in the way described above.

This starting procedure is required due to the hardware design. If one wants to know the background one should continue (cf. Fig. 3.2). But it is not necessary to know all these details for float operations.

During operation the 2 minute strobe is supported at the output of the divider circuit "RED 60", PIN 1. The microprocessor only counts the positive transitions of this 2 minutes strobe:



Once the float battery power is turned on, the circuit RED 60 gets a RESET signal at pin 2. The strobe at pin 1 starts with a low level, that means, after 1 minute the first positive transition occurs and subsequently every 2 minutes:



The clock quartz is constantly powered by the separate 1.5V AA-cell. Pin 5 is the signal input of divider RED 60. Here we measure a continuous 2 second strobe, which cannot be influenced from the outside. The result is a 2 second uncertainty in the 2 minute strobe. Fortunately it is possible to measure this time offset with the FORTH word OFFSET.

5.2 Offset

As described above the time offset (see also FORTH glossary) of the float clock versus the real time clock has to be determind after removing the magnet of the float. This has to be done prior to launching a float. To run "OFFSET", one has only to type in OFFSET and <return> (an

instructional text is shown on the monitor). Hit any key on an even minute of the real time clock. Measure the time the release light is on, or between the beeps. This measurement shows if the float clock is up to two seconds too late or too early.

5.3 Mission Modes

After the float is powered up and the time is adjusted, it is ready to go on a mission. There are two possibilities for missions: a float-supported default mission or a custom designed mission.

5.3.1 Default Mission

Once the float is supplied with power various mission-related variables are initialized. For detailed information refer to FORTH glossary, "DEFAULT".

The Listening Time Table "LTT" contains the daily schedule of the start times (UTC, Universal Time Coordinated) of the listening windows in a 24h cycle:

Default LTT	00:30	-> listen to sound source #1
	01:00	-> listen to sound source #2
	01:30	-> listen to sound source #3
	08:30	-> listen to sound source #1
	09:00	-> listen to sound source #2
	09:30	-> listen to sound source #3
	16:30	-> listen to sound source #1
	17:00	-> listen to sound source #2
	17:30	-> listen to sound source #3

A listening window is the time during which the float listens to a sound source. The listening duration is stored in the variable Listening Length "LLEN", which nominally is 25 minutes. With the default LTT the float

listens to the first sound source at 00:30 for 25 minutes, stores the result in RAM, listens to the second sound source at 01:00 for 25 minutes, stores the result in RAM, and so on.

After every listening window a pressure measurement is taken. If a float sinks deeper than the preset maximum pressure, the mission is aborted and the emergency release is activated. The variable "BOTTOM" contains the preset value for the emergency release. It is set to 1200 dbars by "DEFAULT". Additionally the battery voltage is checked after every window. If the voltage drops below 10V the mission is aborted and emergency release is activated. If a mission is disrupted as explained above the reason is stored in the status byte of the data set (see Section 5.6.).

After every third window of the LTT a temperature and pressure measurement are taken and stored in RAM. This means that temperature and pressure are measured at 01:55, 9:55 and 17:55. If the float is sent on its mission at 17:16, the first listening window starts at 17:30 and the first temperature and pressure measurement (after the emergency pressure check) are taken and stored (see also data storage format, Section 5.6).

The mission length depends on the total number of navigation windows and the number of navigation windows per day. In default mode there are 9 windows per day and a total of 270 navigation windows. This results in a mission length of 30 days. The method of altering mission length is shown in the following example

query no. of windows :	" WINDOWS ? "
	" 270 "
enter new no. of windows :	" 405 WINDOWS ! "
	" WINDOWS ? "
	" 405 "

The new mission length is now (with the default LTT) 45 days. Another way to change the mission length can be achieved with a new LTT. This is described in the next section on custom-designed missions.

After the last window is closed, the emergency pressure is checked and (if the schedule requests) temperature and pressure are measured and the drop weight is released. The duration of the release current is limited to 30 min. After this duration the ARGOS transmission starts at the next half or full hour. Within the next three hours the first data should be available from the ARGOS computer.

5.3.2 Custom Designed Missions

Often it may be necessary to create a new Listening Time Table. The LTT is stored from address $3C80 to $3CF8, i.e. 122 bytes. Since every start time needs 2 bytes for storage, there are 61 addresses for start times of listening windows. To cover 24 hours with continous listening, the highest resolution results in 30-minute-windows. That gives 48 windows (and therefore 48 addresses) per day. For example, 15 minute windows would result in 96 windows per day for continous listening over 24 hours. Since, as mentioned above, there are only 61 addresses left for the storage of the beginning of a listening window, the other 35 times are stored in the other addresses following $3CF8. But in these addresses other mission-related variables are stored. Since the FORTH code does not supply any error message like "access violation" or similar, the float would go into an undefined state because of the overwritten variables. In this case, the mission would not be successful. To summarize, the highest resolution with a continous listening over 24 hours can be achieved with 48 listening windows per day.

It is possible to define a LTT with a higher resolution. Since everything in the FORTH code happens with the positive transition of the 2 minute strobe, the highest resolution would be a new listening window every two minutes. As explained above, there are only 61 addresses for different start times of windows, a mission with 61 2-minute windows would last two hours and 2 minutes. Of course any number larger than 61 can be identified for the windows. However, after the 61st window the next start time would be again the time of the first window of that LTT. The float would go into a "sleep mode" for 21 hours and 58 minutes. Then the 62nd window would start and every two minutes the following window would be activated, if the number of windows was larger than 61.

When writing a new LTT one important thing must be taken into consideration. Since the float clock is incremented only every two minutes the start time of a listening window must be always an even time. Otherwise, all windows with odd start times would start one minute earlier than expected.

5.4 Mission Length

The duration of a complete mission is always the quotient of the total number of listening windows and the Listening Time Table (= windows per day). But there are two restrictions which must be taken into consideration: battery capacity and data storage capacity. As mentioned above, after every third window pressure and temperature measurements are taken. Three windows with a temperature and a pressure measurement are called a measurement cycle, which is 22 bytes long (see section 5.6). One such measurement cycle needs around 7 mAh from the 15V battery pack. In the sleep mode the electronics needs only around 0.8 mA. With the default Listening Time Table (9 windows/day = 3 measurement cycles/day) we estimate a daily energy requirement of

$$3 * 7 \text{ mAh} + 20 \text{ h} * 0.8 \text{ mA} = 37 \text{ mAh}.$$

This represents a power consumption of

$$(0.037 \text{ Ah} \cdot 15\text{V})/24 \text{ h} = 23 \text{ mW}.$$

Using D-cells, which provide >8 Ah (cf. section 3.3.1), the total mission length could last 216 days. If the LTT is reduced to one measurement cycle per day, the battery would last around 320 days.

Data storage capacity must also be taken into consideration. After a mission is completed, the FORTH code prepares the ARGOS messages for satellite transmission. This is done by dividing the total number of stored data bytes by 30. The result is the number of messages. Subsequently a buffer is prepared, where the consecutive numbers of messages to be sent are stored in a random order. Since the numbers in that buffer are stored as 1 byte integers the largest number could only be
255 = $FF. This results in

$$255 \text{ messages} * 30 \text{ bytes/message} = 7650 \text{ bytes}$$

as a maximum of data bytes which are stored. The default LTT supports 3 measurement cycles, which results in 66 bytes per day. Hence the mission length will be:

7650 bytes / 66 bytes/day = 115 days.

To summarize, the maximum number of ARGOS messages can be a limiting factor for practical mission lengths. Reducing the LTT to one measurement cycle per day a mission could last

7650 bytes / 22 bytes/day = 347 days.

Unfortunately this mission length can not be completely covered by the D-cell capacity. In this case one would have to consider the usage of lithium batteries.

5.5 Initializing for Launching

To prepare a float for its mission, it is not nessecary to have detailed knowledge of FORTH-programming. Just a few FORTH words and how to query a word and to set it to a new value must be known. An example of which words must be known is shown by the launching form shown in Table 5.1.

Once a terminal is connected to the float and the main power is supplied by removing the magnet, a red LED on the digital board is turned on and the float prompts with the following logo:

```
RAFOS  FORTH V2.2
A TEAM ROSSBY PRODUCTION
(C) EVERETT CARTER 1988

Float  ID: 508D   The time is: 0

Hit any key before the light goes out to get to FORTH
Otherwise I will start the mission.
Don't  unplug me before I tell you to.
```

RAFOS Auslegeprotokoll

IfM ID: 24 HEX C5548 ARGOS 12629

ARGOS Sender gecheckt J (J/N)
~~Blitzlampe~~ PE=62.468, 62.442 (J/N)

Cruise ID POS 42-4
Datum 25.5.1991

Startdatum 25.05.91

Startzeit 17:33 (Ungerade Minute)

Offset 0.81 s früh

Capture filename c:\float\rafos24.str

Mission: c:\float\bogutach.mis
LLEN 1500.0 sec
WINDOWS ~~4~~ 522
LTT 0030, 0100, 0130 +4 hrs, +8 hr, +12, +16, +20 hrs.
BOTTOM 1200.0 dbar

1. WINDOW Datum 26.05.91
1. WINDOW Zeit 0030 - 0055
Letztes WINDOW Datum 23.6.91 (So.)
Letztes WINDOW Zeit 21:30 - 21:55
Sendedatum 23.6.91
Sendezeit 22:30

Prüfung:

Heartbeat J (J/N) 26 5 91 13:44 Nein 22:12 ✓ 10:58 ✓ 23:01
Auslösenstrom 286 mA
Spannung an Analogbord _____ V

Vor Ausetzen:

Schrauben festziehen _____ (J/N)
Heartbeat J (J/N)

Ausetzen:

Position: 38°35.317' N 17°21.948' W
Datum: 27.05.91
Zeit: 0025 GMT
~~XBT~~/CTD# 110
statim # 557

Ziel:

D 750 T 12.528 S 36.265
Dichte 1.03080

Zusatzgewicht:

naß 340.2 g
trocken 391.2 g
390.0 + 0.3 + 0.9

Bermerkungen:

Auslöser Patron:
Lang / Kurz
Verzüp hell / schwarz

NB1 corrections (-7dbar, +0.05 %) taken into account ✓

launched w/# 45, # 22, # strk drifter 12275

Table 5.1 Launch form example for RAFOS operations on FS POSEIDON.

Now any key must be hit (the red LED turns off) to get to FORTH otherwise mission-related values are set to default values and have match the current purposes. The FORTH prompt is "OK". First query and set the float clock:

```
OK
TIME? 0  OK
1023  Z
TIME? 1022  OK
TIME? 1024  OK
```

TIME? is the only RAFOS FORTH word which is defined without a space between the letters and "?". Now take a temperature and a pressure measurement to check if the thermistor and the pressure transducer are working. This is done with "PP":

```
 OK
PP 2229  113
 OK
```

The first value is centidegrees and the second is centibars. Now set the total amount of navigation windows for the mission:

```
WINDOWS  ? 405  OK
333 WINDOWS !  OK
WINDOWS  ? 333  OK
```

Now ask for the listening length LLEN and set it to a new value:

```
 OK
LLEN ?  15000  OK
10000 LLEN !  OK
LLEN ?  10000 OK
 OK
```

The time for the listening length is stored in deciseconds.

Before launching the float one should make sure that the release current is supplied to the release plug at the end of the mission. This can be

tested with:

```
  OK
LON  OK
LOFF  OK
```

After LON the red light is turned on (this is mentioned in the start logo) and the release voltage is supplied to the release plug. The current can be measured now. After the check turn off the release voltage with LOFF.

BOTTOM is a variable which contains the depth which is chosen as the emergency release depth. This value is expressed in centibars.

```
  OK
BOTTOM ? 12000  OK
11111 BOTTOM !  OK
BOTTOM ? 11111  OK
  OK
```

One last synchronization check follows. This is done by calling OFFSET and executing the necessary instructions (see Sections 5.1 and 5.2).

```
  OK
OFFSET
To determine the clock offset. Hit any key on an
even minute. Time the elapsed time the release light
is on, or between the beeps.
 Ready Done
 OK
```

On an even minute (real time) any key is pressed and a stop watch is turned on. The red release LED on the digital board is activated. This procedure ends with the next heart beat. This should happen with the next even minute. But hardware characteristics (see Section 5.1) give the float clock an uncertainty of some fractions of a second, which has to be measured.

Assuming that the Listening Time Table should be taken unchanged from the default mode (cf. section 5.3.1) the float can now be sent on its mission. This is simply done by typing:

```
OK
MISSION
```

The float responds with:

```
Float ID: 508D   The time is: 1626

Hit any key before the light goes out to get to FORTH
Otherwise I will start the mission.
Don't unplug me before I tell you to.
OK you can unplug me now.
Window  TOD Time Cor
0 1630  1654 3311 83
1 1700    < etc. >
```

The float gives the order to unplug the terminal. It is a good idea to leave the float connected to the terminal for at least one navigation window to verify proper operation.

5.6 Data Storage Format

From address $2000 in RAM B a RAFOS float ($044C in RAM A for MAFOS monitor) stores its measured data. With the FORTH-word "READRAM" (to be downloaded prior to use), it is possible to dump the RAM-contents to a log file. An example is shown in Tab. 5.2 for float $5D94. For clearity, only the first 244 bytes are shown.

```
 OK
READRAM
RAM - Inhalt von Adresse : 2002  bis Adresse : 2210
Float - ID : 5D94
Status-byte : 0   Anzahl der messages : 12

24  0 DD  0  0 21   27  1  8 18  0 CA   20  1 1D 16  0 1C   9 6A   2 22
20  0 F9 17  0 95   1B  1 3F 19  0 38   1B  0 6D 17  0 1A   7 6A   6 40
1D  1 66 1C  0  A   1D  0 9C 19  0 22   1F  1 DA 1D  0 38   6 86   A  A
1E  0 5F 19  0 1F   22  1 D2 1D  1 75   1F  0 A3 19  0 5B   5 EE   D 13
1C  0 83 18  0 14   21  1 37 18  0 18   24  2 10 20  0 51   5 81  10  1
21  1 1A  0  1 21   1C  2 10 1A  0 12   1E  1 94  0  2 10   5 42  12 A3
26  0 A2 1D  0 6C   1E  2 20 1D  0 58   26  1 D1 1F  0 5B   5 14  15 22
1E  1 97 19  0 6C   1D  2 2F 19  0 1F   1C  0 69 15  0 14   4 F4  17 12
20  0 54 16  0  5   1F  2 34 1D  0 56   1E  0  7  0  2 34   4 DF  18 1E
21  0 56 19  0 22   21  0  2  0  0 56   1A  0 57  F  0  5   4 DC  18 DF
1F  1 BD 1D  0 4C   20  0 22  0  1 BD   21  1 D4 1A  0 A7   4 CC  19 AE
```

Table 5.2 RAM contents of float 5D94. All numbers are given in hexadecimal. Source code of READRAM.4TH is reproduced in Appendix A1.

The first byte in address $2000 is a status byte. It can have only one of three states:

1) $0: the complete mission worked well.

2) $50: pressure failure ($50 = ASCII 'P'). The float has gone too deep. The mission was canceled and the float has made an emergency release.

3) $42: battery failure ($42 = ASCII 'B'). The battery voltage dropped below 10 V. The mission was canceled and the float made an emergency release.

The second byte at address $2001 is the total number (rounded upwards) of 30-byte segments or ARGOS messages (here $12) in RAM. Since the number of total ARGOS messages is stored in only a 1-byte number, $FF = 255 ARGOS messages is the maximum for the mission length. The remaining bytes are all numbers concerning the float mission. These bytes are subdivided in navigation packets, temperature and pressure values. For example see line 9 of Tab. 5.2. The first six bytes belong to a group called navigation packet 1, the second six bytes belong to navigation packet 2 and the third six bytes are navigation packet 3. The next two bytes represent temperature and the last two bytes contain the pressure. Tab. 5.3 shows these numbers in more detail.

Cor1	Time1	Cor2	Time2				Cor1	Time1 (10^{-1}s)	Cor2	Time2 (10^{-1}s)
$20	$00 $54	$16	$00 $05	=	NavPack 1	=	32	84	22	5
$1F	$02 $34	$1D	$00 $56	=	NavPack 2	=	31	564	29	86
$1E	$00 $07	$00	$02 $34	=	NavPack 3	=	30	7	0	564
			$04 $DF	=	Temperature	=	1247 centidegrees			
			$18 $1E	=	Pressure	=	6174 centibars			

Table 5.3 Details of line 9 in Table 5.2. Conversion from hexadecimal to decimal numbers is done with DECODRAM.PAS, reproduced in Appendix A2.

One navigation packet contains two "correlation heights" (Cor#) and two "arrival times" (Time#) of the sound signal. The sound source signals are recognized with a hardware filter and the microprocessor compares the received signals to the expected one. The time of arrival (travel time of incoming signals) is measured with respect to the beginning of each listening window. This arrival time is referrenced to the end of the detected signal. According to the match, each travel time is associated with a relative correlation height, which is a measure of signal strength for quality control. In navigation packet 1 the first byte ($20) is the largest correlation height found during the previous navigation window. The following two bytes ($00 $54) are the time of arrival of the sound signal when this correlation height was found (in deciseconds). The fourth byte ($16) is the second largest correlation height from the previous navgiation window. Bytes five and six are the corresponding time of arrival for this correlation ($00 $05). The next two navigation packets are organized in the same way. Bytes 19 and 20 are the temperature and bytes 21 and 22 are the pressure. After the pressure measurement the float continues with the next navigation window. So three navigation windows with one temperature measurement and one pressure measurement is 22 bytes long. It is called one "measurement cycle".

5.7 Data Transmission Format

Data are transmitted according to the ARGOS data format specifications. The strict formating is software-controlled by the float microprocessor, which makes a separate data controller as part of the transmitter obsolete. The ARGOS data structure is reproduced in Tab. 5.4, taken from the Telonics

ARGOS DATA FORMAT/STRUCTURE

```
Bits:(1111111 11111110 00101111 nnnpiiii iiiiiiii iieeeeee)+{4 to 32}
        15          8       1   3 1       14          6      { BYTES }

        B           F       I   N E       I           I          D
        I           O       N   U V       D           D          A
        T           R       I   M E                              T
                    M       T   B N       N           N          A
        S           A       I   E         U           U
        Y           T       A   R P       M           M
        N                   L     A       B           B
        C           S       I   O R       E           E
                    Y       Z   F I       R           R
                    N       A     T
                    C       T   G Y                   E
                            I   R                     C
                            O   O O                   C
                            N   U N
                                P                     C
                                S G                   O
                                  R                   D
                                  O                   E
                                  U
                                  P
                                  S
```

NOTE: one "GROUP" = one "ARGOS WORD" = four (4) data bytes.

Table 5.4 ARGOS data format and structure taken from the Telonics receiver manual. The bar denotes the 20 bit long ID number, discussed in Fig. 4.1a.

6. Sound Source Moorings

As we briefly explained in section 2, sound sources are an integral part of the RAFOS system. Coded sound waves (cf. 3.3.2) are transmitted typically three times per day and then are heard by drifting RAFOS floats. The exact time of arrival of these signals at the float location is the basis for all navigational computations with float data.

Sound sources used by IfM Kiel are commercial products of Webb Research Cooperation Inc. of Falmouth, MA, USA (Fig. 2.3). For operational details we refer to the sound source manual. Prior to launching, synchronization is done by a radio controlled PC clock. The personal computer is used in a terminal emulation mode for programming the internal microprocessor of the source. The transmission times used to set the source are for the start of the signal, whereas the arrival time recorded by the float is the end of the signal. This must be adjusted in the tracking software.

In Fig. 6.1 we show a typical mooring design as presently (1991) used in the Iberian Basin (cf. Fig. 2.4). As anchors we use 2 - 3 railway wheels. The mooring line (11 and 14 mm, Perlon "Meteor"-Leine) are interrupted by the acoustic release (Oceano Instruments, type RT 161), glass balls (Benthos Inc., 17'), and the sound source (WRC). To protect the rope from intolerable torques we use swivels (NSW) beneath and above the sound source. The top bouy is equipped either with a short wave radio transmitter (IBAK) or an ARGOS "Watch Dog" transmitter (IfM Kiel).

After a deployment time of two years we detected a severe corrosion problem on the aluminum (6016) sound sources. We are cooperating closely with Mr. D. Webb from WRC to solve these problems for future use of recovered instruments.

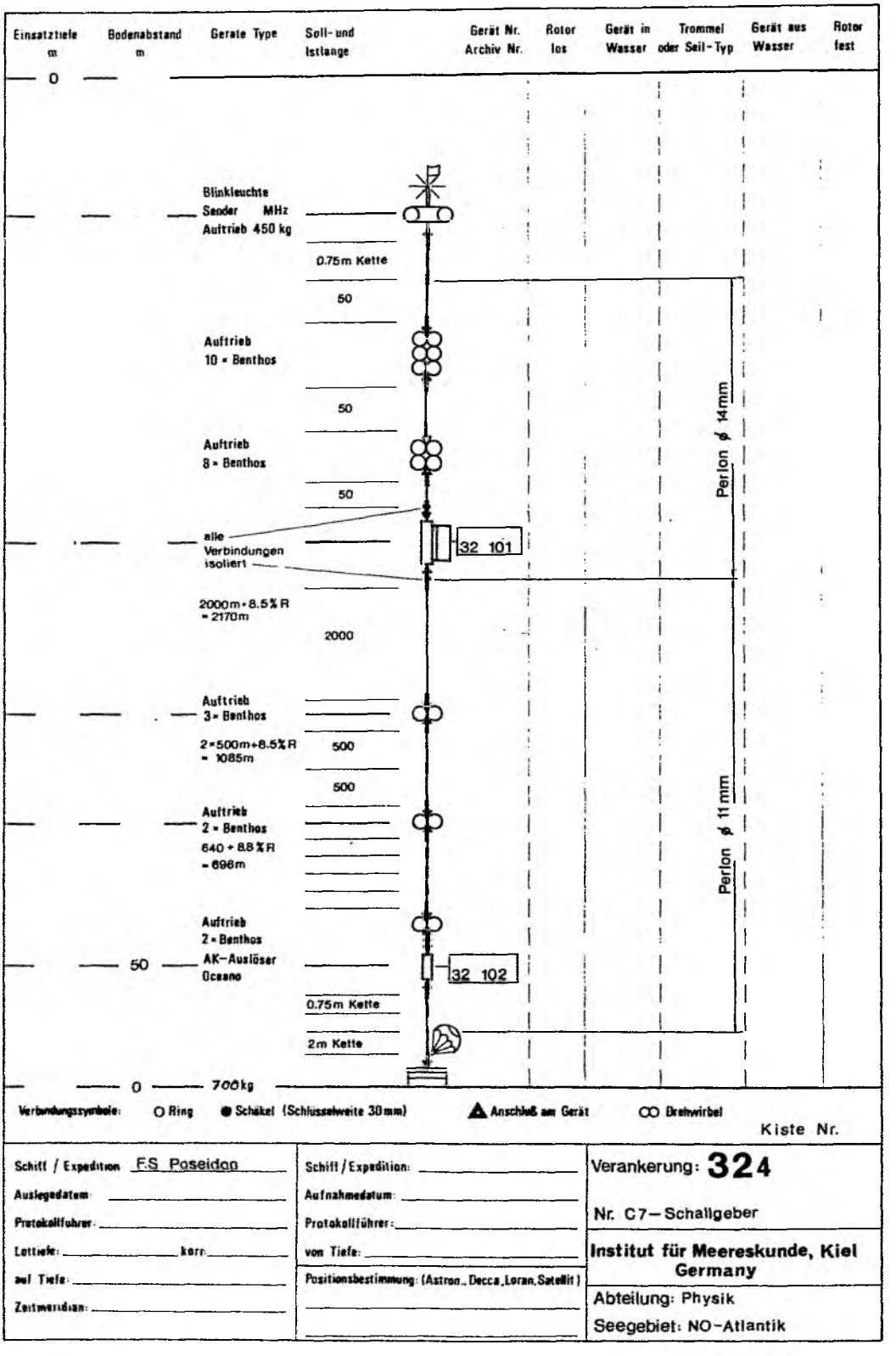


Fig. 6.1 Typical mooring for a sound source in the Iberian Basin.

7. Spin-off Instruments

7.1 MAFOS Monitor

The primary aim in the development of the MAFOS (Moored rAFOS) instrument was to definitively determine the timing accuracy (or operation) of three RAFOS sound sources moored in the Iberian Basin. The secondary purpose of the MAFOS was to test the "float" performance itself (König et al., 1991).

There are two fundamental differences between a RAFOS float and a MAFOS monitor. First, a MAFOS, which is placed on a recoverable deep-sea mooring, does not need an Argos transmitter or antenna. Second, the ballast release mechanism of the RAFOS float is unnecessary. A MAFOS monitor is shown in Figure 6.1. It is housed in a borosilicate glass pipe nearly identical to that used for RAFOS floats. However, it is only 70 cm long, i.e., about half the length of a float, as described in section 3. Unlike the RAFOS float, RAM A can be used for additional data storage. This results in a total memory capacity of almost 14500 bytes. For details on the memory map see Fig. 3.3 and for possible mission length see section 5.4:

$$(\$3C80-\$1)_{RAM\ B} - (\$03EE + \$1)_{RAM\ A} = \$\ 3890 = 14480$$

However, for practical purposes we reduced the starting address in RAM A from $03EE to $044C (cf Fig. 4.1c).

7.2 CB-Float

A second very useful development is the CB-Float. It is nearly identical to the RAFOS float (cf Fig. 2.2) except that the ARGOS transmitter and antenna are replaced by a 27 MHz ("CB-frequency") transmitter, an adequate antenna, and an additional flash light.

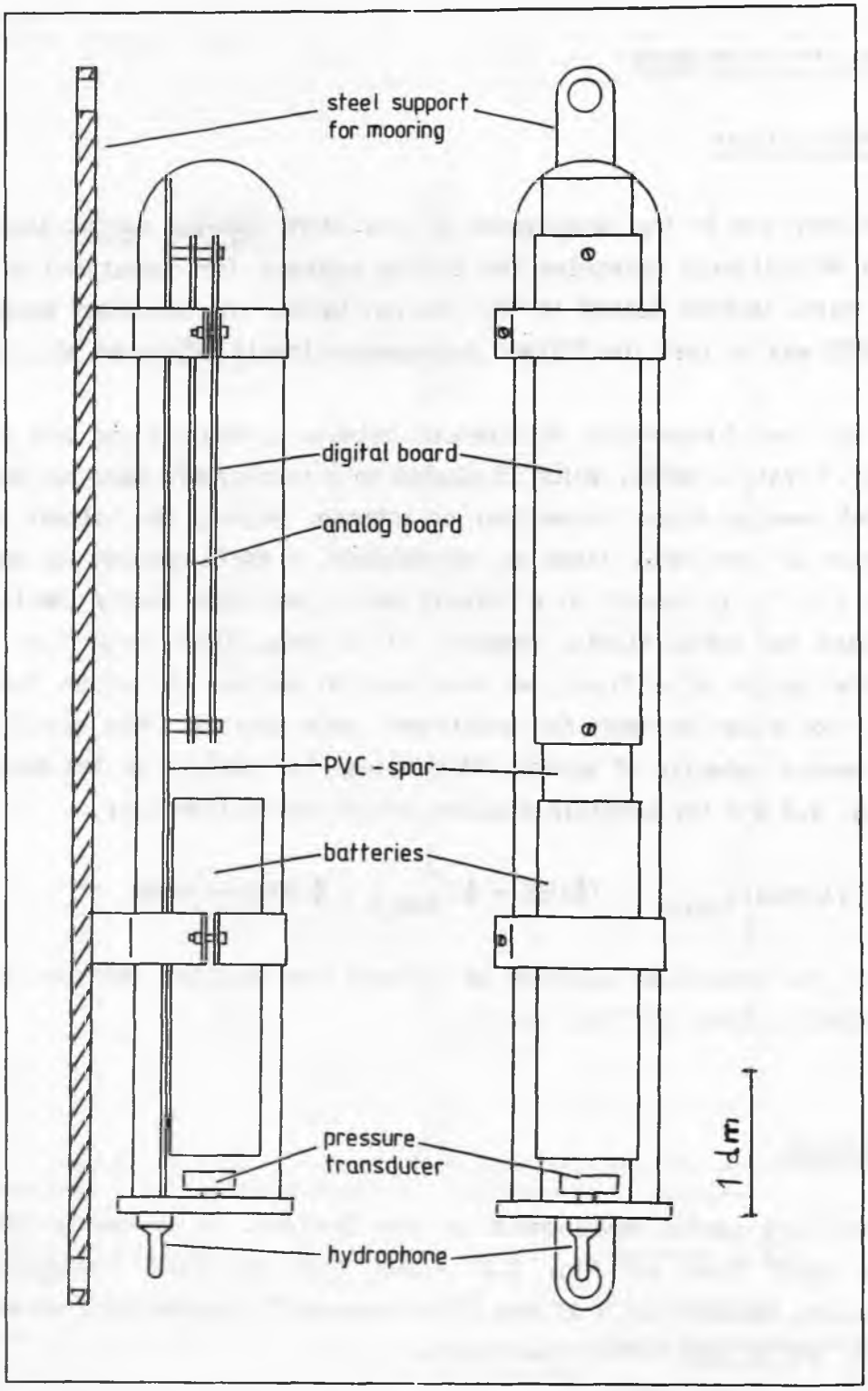


Fig. 7.1 MAFOS monitors act like moored "floats". They need no transmitter and no release plug (cf. Fig. 3.1). They are used for control purposes of the sound array (cf. Fig. 2.4).

Fig. 7.2 MAFOS monitors are protected by a PVC tube. The displayed device was recovered on board FS POSEIDON in May 1990 (cf. Table 1.1).

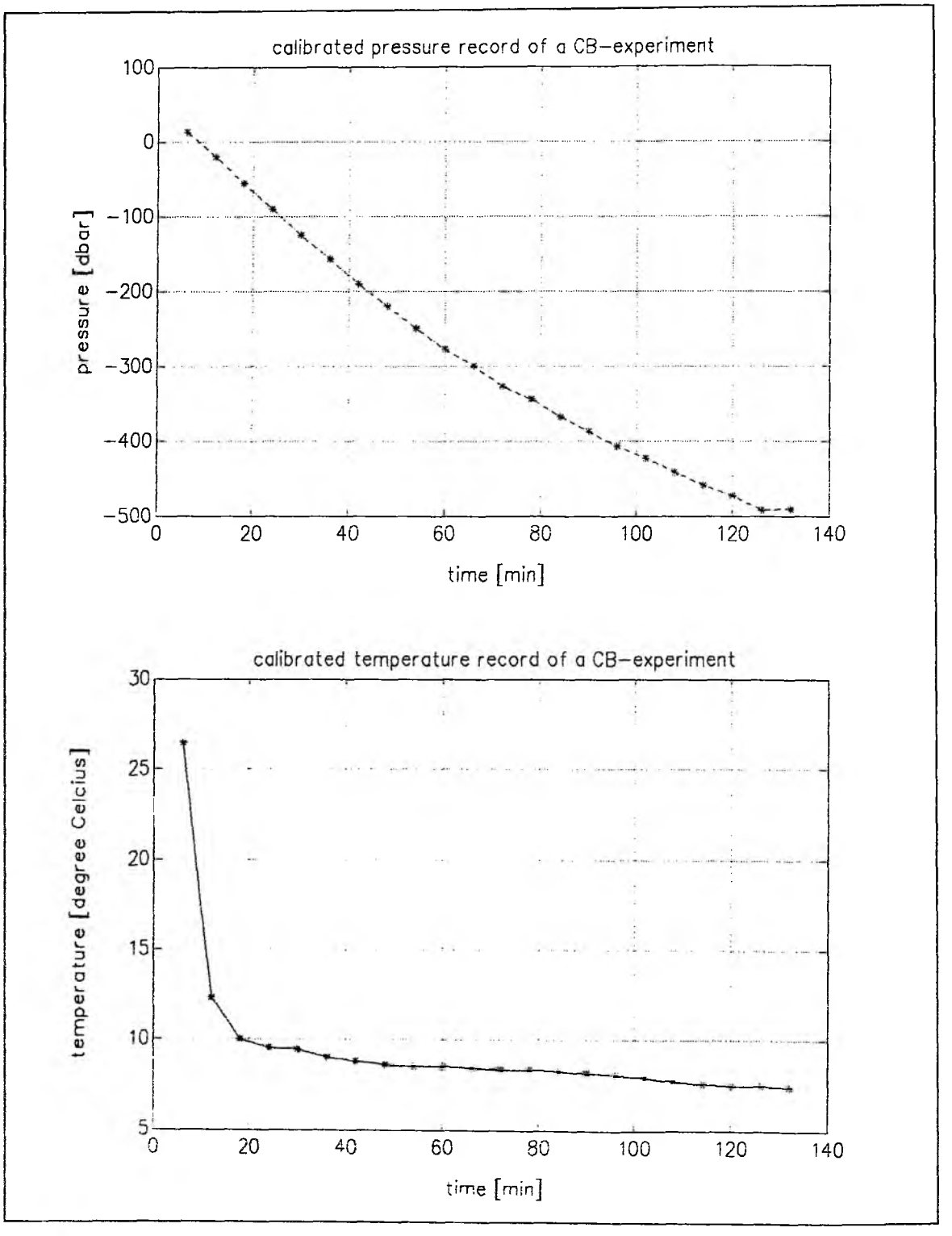


Fig. 7.3 Typical sinking record from CB floats as obtained during POSEIDON cruise 172/5 (cf. Table 1.1). The top graph shows the approach to the target pressure of 700 dbar. The associated temperature record is shown on the bottom.

The purpose of the CB-float is to check our ballasting results from the pressure tank under in situ conditions. The FORTH code allowed for an easy programming of a new Listening Time Table. Directly after recovery of the CB-float its memory content is dumped on a personal computer, where data are made available for analysis. After the release plug is changed, the CB-float is ready for a subsequent experiment. The battery capacity lasts for at least ten CB—float deployments. Since everything in the float occurs in 2-minute strobes, 2 minutes was the shortest time for navigation windows. There are 61 bytes left in RAM B for different listing times. This results in a maximum mission length of 112 minutes with a 2-minute resolution of a temperature and pressure record (Fig. 7.3).

Best recovery results with CB-floats were obtained at night. In spite of the flasher, the float can hardly be located during the day. During one night experiment the flasher of a CB-float was seen at a distance of 5 nautical miles.

8. Subsequent Data Flow

A short overview of the RAFOS data processing is given in Fig. 8.1. This program package originally was developed by T. Rossby and his team at the University of Rhode Island, Graduate School of Oceanography. Portions of these programs were implemented on a main frame (DEC, VAX) as well as on a personal computer (IBM, PS2) at IfM Kiel.

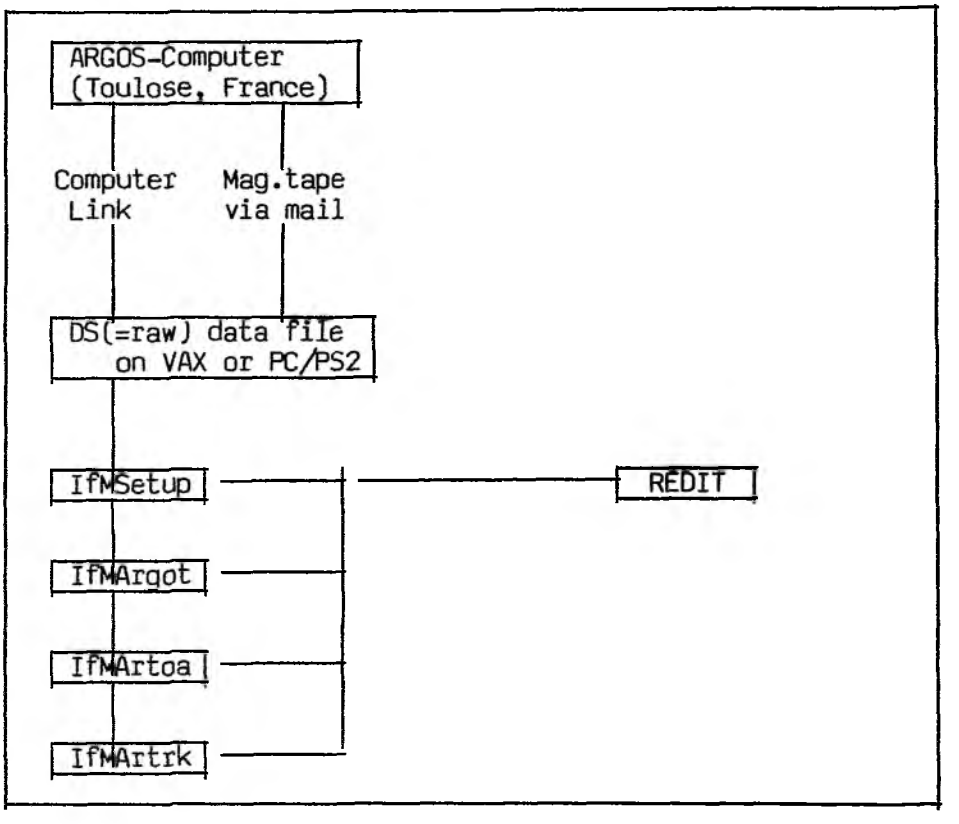


Fig. 8.1: Flow chart of RAFOS data processing.

Once float data arrive in Kiel, they are processed in four different steps. A float inventory and four other computer programs are necessary to obtain trajectories, velocities, and calibrated temperature and pressure data.

* Float inventory

REDIT creates for every float a separate data file, where all float and specific data are stored: ID number; date, time and position of launching; individual calibration coefficients for temperature and pressure sensors; etc.

* Initial data processing

IfMSetup sorts incoming raw data according to all float ID numbers. For each float a binary raw data file is created, where all 32-byte ARGOS messages are stored.

IfMArgot unscrambles raw data. All hexadecimal data are checked for redundancy and proper satellite transmission. Data counts are stored in decimal form in an outputfile.

IfMArtoa is the calibration programm. It converts all counts into oceanographic quantities: time of arrival of the sound signal in seconds, temperature in °C and pressure in dbar. Individual coefficients from REDIT are applied.

IfMArtrk calculates float trajectories. The output file contains date, temperature, pressure, float position, horizontal and vertical velocities. Provisions are made for signals from two or three sound sources in the same listening window. Lost data can be interpolated. The output data are ready for plotting (Fig. 8.2) and scientific analysis.

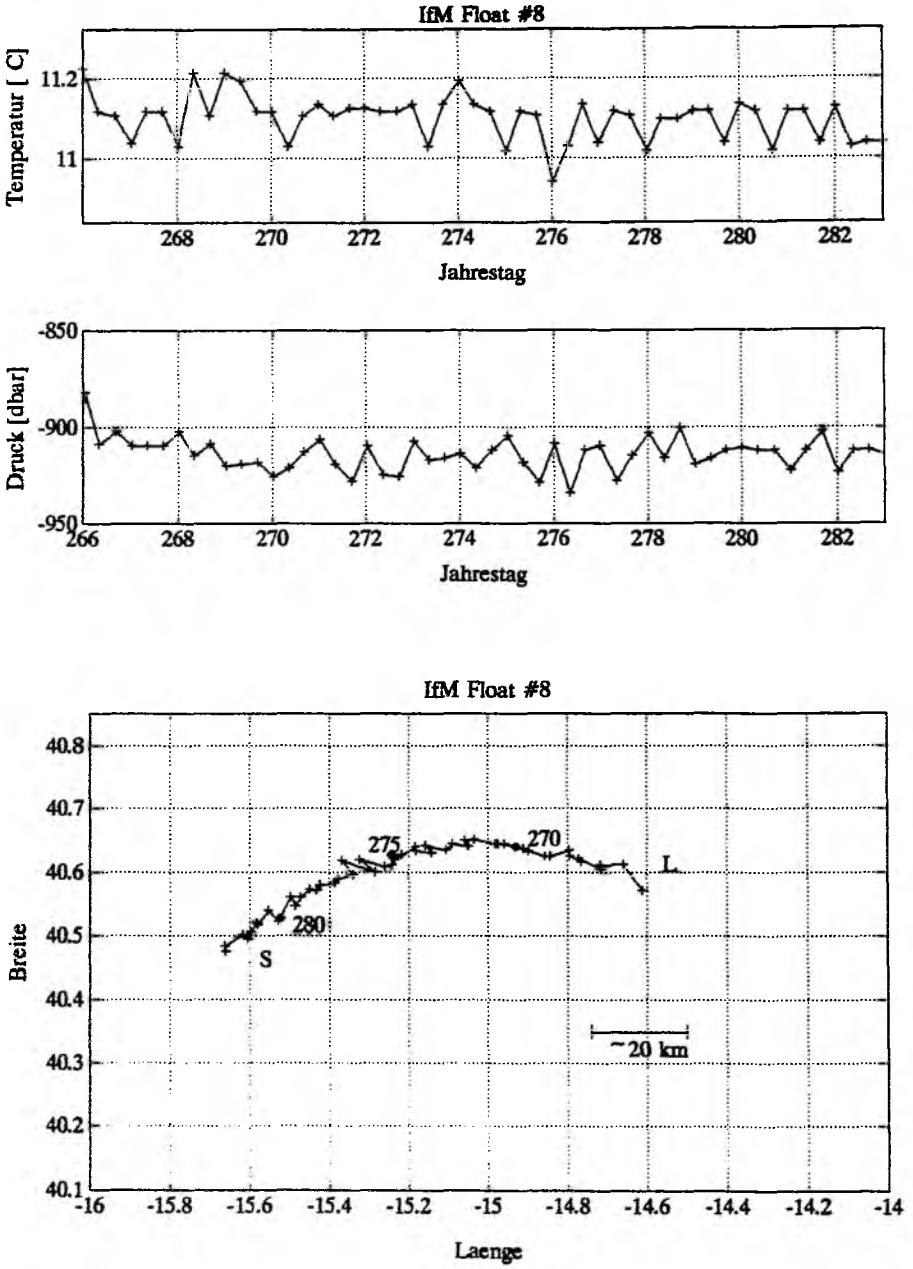


Fig. 8.2 Example for processed float data from the test phase in the Iberian Basin. Float #8 was launched from FS METEOR in October 1990 (cf. Table 1.1). On top and in the middle records of temperature and pressure are presented as a function of year days. Day numbers are reproduced in the trajectory plot (bottom), where launch (L) and surface (S) locations are included.

9. Acknowledgements

First and foremost we thank Tom Rossby who made his RAFOS technology available to us. It took at least a year after RAFOS floats were introduced in Kiel before commercial versions came on the market. The close cooperation with E. Carter and J. Fontaine from the Graduate School of Oceanography at Rossby's laboratory and with D. Dorson (Bathy Systems, Inc.), D. Webb (WRC, Inc.), and P. Tillier (formerly at WRC, now at SeaScan Inc.) are gratefully acknowledged. Not forgotten is the excellent cooperation with local German companies and their representatives: Herr Rosenblatt (Schröder and Co.), Herr Inselmann (Jenaer Glas), Herr Fischer (Hydrobios) and many others. Repeated encouragement was provided by R. Käse, W. Krauß and G. Siedler. We express many thanks for the help to all members of the float group in Kiel: U. Huenninghaus who handles all mechanics, P. Meyer and M. Möllenhoff who are in charge of the float electronics, and – last but not least - K. Schultz Tokos who is involved in float ballasting and in the analysis of float data. All sound source moorings are managed by D. Carlsen.

Special thanks go to the captains and crews of the research vessels POSEIDON, METEOR, and SONNE. In connection with these activities we acknowledge the help of B. v. Bodungen (IOF, Warnemünde), J.-C. Gascard (L.O.D.Y.C., Paris), G. Hesloin and N. Cortez (both IFREMER, Brest), Director Biscoito (Municipal Museum, Funchal), Cptn. Cordoso (Hydrographic Office, Lisboa), H. Kudraß (BGR, Hannover), and R. Zahn-Knoll (GEOMAR, Kiel). A. Bower and B. Owens (both WHOI, Woods Hole) provided valuable suggestions and greatly improved the English text. The lay-out of the manuscript was thoughtfully prepared by S. Drews, M. Vanicek and P. Goldschmidt.

The introduction of the RAFOS technology in Kiel would not have been possible without the significant financial support of the Deutsche Forschungsgemeinschaft, Bonn (SFB 133).

10. References

Carter, E. (1986): Forth goes to sea. Dr. Dobb's Journal Software Tools, 117, 40-44.

Carter, E. (1989): pers. communication.

Gray, K.O. and J.D. Stachiv (1969): Light housings for deep submergence applications - Part III: Glass pipes with conical flanged ends. Naval Civil Engineering Laboratory Rep. R-618.

Kelly, M.G. and N. Spies (1986): FORTH, a text and reference. Prentice Hall, Englewood Cliffs, 487 pp.

König, H., K. Schultz Tokos and W. Zenk (1991): MAFOS: A simple tool for monitoring the performance of RAFOS sound sources in the ocean. J. Atmos. Oceanic Technol., 8, 669-676.

Krauss, W. and R.H. Käse (1984): Mean circulation and eddy kinetic energy in the eastern North Atlantic. J. Geophys. Res., 89(C3), 3407-3415.

Krauss, W. (1986): The North Atlantic Current. J. Geophys. Res., 91, 5061-5074.

Richardson, P.L. (1991): SOFAR floats give a new view of ocean eddies. Oceanus, 34, 23-31.

Rossby, H.T., D. Dorson and J. Fontaine (1986): The RAFOS System. J. Atmospheric and Oceanic Technology, 3(4), 672-679.

Rossby, H.T., E.R. Levine and D.N. Connors (1985): The isopycnal Swallow float - A simple device for tracking water parcels in the ocean. Progress in Oceanography, 14, 511-525.

SFB 133 (1991): Arbeits- und Ergebnisbericht 1989-90-91, Sonderforschungsbereich 133 "Warmwassersphäre des Atlantiks". Chr.-Albrechts-Uni., Kiel, 282 pp.

Stommel, H. (1955): Discussion at the Woods Hole Convocation, June 1954. J. Mar. Res., 14, 504-510.

Swallow, J. (1955): A neutral-buoyancy float for measuring deep currents. Deep-Sea Res., 3, 74-81.

VARTA (1979): VARTA-Gerätebatterien, Technisches Handbuch Primärbatterien. VARTA Batterien AG, Hannover.

Webb, D.C. (1977): SOFAR floats for POLYMODE. Oceans '77 MTS-IEEE Conf. Proc., 2, 44B, 1-5.

Young, W.C. (1989): Roark's formulas for stress and strain. McGraw-Hill Inc., 6th edition, 763 pp.

Zenk, W. (1990): Project "German RAFOS", ARGOS Newsletter, 39, 13-17.

Appendix A1

READRAM.4TH

```
HEX
VARIABLE COUNTER
: READRAM                  ( Version 23. JULY 1990 , f. neue ROM's )
CR                         (                Holger K. )
    HEX
    1 COUNTER !        ( Startwert des Zaehlers )
    20C8               ( Endadresse )
    2002               ( Startadresse )

    DUP ROT DUP ROT ( Stack f. Textausg. vorbereiten )

    ." RAM - Inhalt von Adresse : " . ." bis Adresse : " . CR
    ." Float - ID : "  ID . CR
    ." Status-byte :  " 2000 C@ DUP 0= IF . ELSE EMIT  THEN
    ."   Anzahl der messages : " 2001 C@ DECIMAL . HEX
    CR CR
    SWAP
      DO   ( Schleife  f. Ausgabe 22 bytes/Zeile )
        I  ( Schleifenwert v. Returnstack kopieren )
        C@
           DUP 10 < IF            ( Falls RAM-Zelle nur ein  )
                     SPACE .      ( byte , Ausgabe mit Space )
                   ELSE
                     .            ( sonst normale Ausgabe    )
                   THEN
        COUNTER @ 16  = IF
                          1 COUNTER !
                          CR
                        ELSE
                          COUNTER @ 1 + COUNTER !
                        THEN
      LOOP
  DECIMAL
  CR
  ABORT   ( Stack reinigen )
   ;
```

Appendix A2

DECODRAM.PAS

" | " stands for " { "

"ü" stands for " } "

```
PROGRAM DecodRam (InPutFile,OutPutFile);

(*****************************************************************)
(*                                                               *)
(*      Dieses Programm ist im hoechsten Masse unvollstaendig !! *)
(*      Verbesserungsvorschlaege jederzeit willkommen !!         *)
(*      Dies Programm liesst den File ein , welcher von          *)
(*      READRAM.4TH erzeugt wurde . Danach wird ein dekodierter  *)
(*      Outputfile angelegt .                                    *)
(*      Zum Compilieren dieses Programms muss(!) der MATH.TOL-   *)
(*      File in der gleichen Directory vorhanden sein .          *)
(*      Programm ist mit TURBO-PASCAL Compiler Vers. 3.02 A      *)
(*      geschrieben .                                            *)
(*                                                               *)
(*                                   Holger K. , 28. Feb. 1990   *)
(*                                                               *)
(*    01.03.1990 In Procedur Read_InPutFile Abfrage fur neue     *)
(*               ROM's geaendert .        Holger K.              *)
(*    10.05.1990 Fur POSEIDON 172 in Zeile 273 Header und in     *)
(*               in Zeile 311 Tabellenuberschrift fur Output-    *)
(*               file lahmgelegt . Kann bei Bedarf rückgängig    *)
(*               gemacht werden .          Holger K.             *)
(*    17.05.1990  + statt * , pierre                             *)
(*                                                               *)
(*    11.07.1990 PROGRAMM AUF TURBO-PASCAL Vers. 5.0 UMGESTELLT  *)
(*                                                               *)
(*    25.07.1990 In Procedure "Decimal_Bytes_to_Real" IF-Abfrage *)
(*               eingebaut, weil bei Atmospharendruck negative   *)
(*               Drucke auftreten konnen.                        *)
(*                                                  Holger K.    *)
(*    08.03.1991 In Offnen_OutPutFile Option eingebaut, um einen *)
(*               OutputFile zu uberschreiben, falls gewunscht.   *)
(*                                                  Holger K.    *)
(*****************************************************************)

  Uses Crt , Dos;

  Const
        Dummy       = 80 ;


  TYPE

        DatenFile   = Text;
        tEintrag    = STRING[80] ;
        String79    = STRING[79] ;   (* den braucht man f. MATH.TOL *)

  VAR

        OutPutFile        : Datenfile;
        InPutFile         : datenfile;
        Buffer            : STRING[65];
```

```
      OutPutFile_Exist : BOOLEAN ;
| ----------------------------------------------------------Ü

| $I MATH.TOLÜ

| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü


PROCEDURE WaitKey;          | Meldung ausgeben und auf RETURN warten Ü
   BEGIN
     Write(' <RETURN> druecken !');ReadLn;
   END;

| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü

FUNCTION JaNein(Frage : tEintrag ) : BOOLEAN;

  VAR   Wahl : CHAR;                    | Frage solange      Ü
                                        | wiederholen, bis   Ü
  BEGIN                                 | ja oder nein ge-   Ü
    REPEAT                              | antwortet wird.    Ü
      Write(Frage,' ? ');ReadLn(Wahl);  | Wenn "ja", kommt   Ü
    UNTIL ( Wahl IN [ 'j','J','N','n' ]); | .TRUE.  zurueck. Ü
    JaNein := (Wahl IN [ 'J','j']);
  END;

| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü

  FUNCTION FileExist(Name:tEintrag) : BOOLEAN;

    VAR   f : FILE;                     | Ueberpruefen , ob   Ü
                                        | Datenfile schon     Ü
    BEGIN                               | existiert. Wenn ja, Ü
      Assign(f,Name);                   | kommt .TRUE. zurueck. Ü
    (*$I-*)
      ReSet(f);
    (*$I+*)
      FileExist := (IOResult = 0);
    END;
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü
| ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü

  PROCEDURE Decimal_Bytes_to_REAL( Cor11 , Tim111 , Tim112 ,
                                   Cor12 , Tim121 , Tim122 ,
                                   Cor21 , Tim211 , Tim212 ,
                                   Cor22 , Tim221 , Tim222 ,
                                   Cor31 , Tim311 , Tim312 ,
                                   Cor32 , Tim321 , Tim322 ,
                                   Temp1 , Temp2  ,
                                   Press1 , Press2 : REAL );
```

```
VAR
      Time_11 , Time_12 ,
      Time_21 , Time_22 ,
      Time_31 , Time_32 ,
      Temp , Press       : REAL ;

 Begin
      Cor11   := Cor11 * 2 ;
      Time_11 := ( Tim111 * 256 + Tim112 ) * 0.1 ;
      Cor12   := Cor12 * 2 ;
      Time_12 := ( Tim121 * 256 + Tim122 ) * 0.1 ;
      Cor21   := Cor21 * 2 ;
      Time_21 := ( Tim211 * 256 + Tim212 ) * 0.1 ;
      Cor22   := Cor22 * 2 ;
      Time_22 := ( Tim221 * 256 + Tim222 ) * 0.1 ;
      Cor31   := Cor31 * 2 ;
      Time_31 := ( Tim311 * 256 + Tim312 ) * 0.1 ;
      Cor32   := Cor32 * 2 ;
      Time_32 := ( Tim321 * 256 + Tim322 ) * 0.1 ;

      Temp    := ( Temp1 * 256 + Temp2 ) * 0.01 ;

      IF Press1 > 36 THEN  (* Druckw. geht nur bis 14000 cbar *)

                Press := ((Press1 - 256)*256 + Press2 ) * 0.1
            ELSE
                Press := ( Press1 * 256 +  Press2 ) * 0.1 ;

    Write(Cor11:3:0,Time_11:7:1,Cor12:4:0,Time_12:7:1,
          Cor21:5:0,Time_21:7:1,Cor22:4:0,Time_22:7:1,
          Cor31:5:0,Time_31:7:1,Cor32:4:0,Time_32:7:1,
          Temp:6:2,Press:7:1);

    IF OutPutFile_Exist THEN
       WriteLn(OutPutFile,
               Cor11:3:0,Time_11:7:1,Cor12:4:0,Time_12:7:1,
               Cor21:5:0,Time_21:7:1,Cor22:4:0,Time_22:7:1,
               Cor31:5:0,Time_31:7:1,Cor32:4:0,Time_32:7:1,
               Temp:6:2,Press:7:1);
 End;
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü

PROCEDURE StrToHex (s : string79;VAR h : real);
  VAR
        error       : BOOLEAN;
  BEGIN
    IF s [1] = ' ' THEN s := Copy (s,2,1);
    anybase_to_decimal (s,16,h,error);
  END;

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++Ü
PROCEDURE String_2_Values;
```

```
VAR
   InPut_Buffer : Array[0..80] of Byte ;
   BufferStr    : STRING[80] absolute InPut_Buffer;
   Nav_pack1_Cor1 ,
   Nav_pack1_Tim11 ,
   Nav_pack1_Tim12 ,
   Nav_pack1_Cor2 ,
   Nav_pack1_Tim21 ,
   Nav_pack1_Tim22 ,
   Nav_pack2_Cor1 ,
   Nav_pack2_Tim11 ,
   Nav_pack2_Tim12 ,
   Nav_pack2_Cor2 ,
   Nav_pack2_Tim21 ,
   Nav_pack2_Tim22 ,
   Nav_pack3_Cor1 ,
   Nav_pack3_Tim11 ,
   Nav_pack3_Tim12 ,
   Nav_pack3_Cor2 ,
   Nav_pack3_Tim21 ,
   Nav_pack3_Tim22 ,
   Temp_1 , Temp_2 ,
   Press_1 ,Press_2          : STRING[2];

   Cor11 , Tim111 , Tim112 ,
   Cor12 , Tim121 , Tim122 ,
   Cor21 , Tim211 , Tim212 ,
   Cor22 , Tim221 , Tim222 ,
   Cor31 , Tim311 , Tim312 ,
   Cor32 , Tim321 , Tim322 ,
   Temp1 , Temp2  ,
   Press1 , Press2           : REAL ;

Begin
(*      Writeln('String2value : ',Buffer);                *)

   Nav_pack1_Cor1 := Copy(Buffer,1,2);
   Nav_pack1_Tim11 := Copy(Buffer,4,5);
   Nav_pack1_Tim12 := Copy(Buffer,7,8);
   Nav_pack1_Cor2 := Copy(Buffer,10,11);
   Nav_pack1_Tim21 := Copy(Buffer,13,14);
   Nav_pack1_Tim22 := Copy(Buffer,16,17);

   Nav_pack2_Cor1 := Copy(Buffer,19,20);
   Nav_pack2_Tim11 := Copy(Buffer,22,23);
   Nav_pack2_Tim12 := Copy(Buffer,25,26);
   Nav_pack2_Cor2 := Copy(Buffer,28,29);
   Nav_pack2_Tim21 := Copy(Buffer,31,32);
   Nav_pack2_Tim22 := Copy(Buffer,34,35);

   Nav_pack3_Cor1 := Copy(Buffer,37,38);
   Nav_pack3_Tim11 := Copy(Buffer,40,41);
   Nav_pack3_Tim12 := Copy(Buffer,43,44);
   Nav_pack3_Cor2 := Copy(Buffer,46,47);
   Nav_pack3_Tim21 := Copy(Buffer,49,50);
   Nav_pack3_Tim22 := Copy(Buffer,52,53);
```

```
        Temp_1 := Copy(Buffer,55,56);
        Temp_2 := Copy(Buffer,58,59);

        Press_1 := Copy(Buffer,61,62);
        Press_2 := Copy(Buffer,64,65);

(*       Convert parts of string to REAL values   *)

        StrToHex (Nav_pack1_Cor1,Cor11);
        StrToHex (Nav_pack1_Tim11,Tim111);
        StrToHex (Nav_pack1_Tim12,Tim112);

        StrToHex (Nav_pack1_Cor2,Cor12);
        StrToHex (Nav_pack1_Tim21,Tim121);
        StrToHex (Nav_pack1_Tim22,Tim122);

        StrToHex (Nav_pack2_Cor1,Cor21);
        StrToHex (Nav_pack2_Tim11,Tim211);
        StrToHex (Nav_pack2_Tim12,Tim212);


        StrToHex (Nav_pack2_Cor2,Cor22);
        StrToHex (Nav_pack2_Tim21,Tim221);
        StrToHex (Nav_pack2_Tim22,Tim222);
        StrToHex (Nav_pack3_Cor1,Cor31);
        StrToHex (Nav_pack3_Tim11,Tim311);
        StrToHex (Nav_pack3_Tim12,Tim312);

        StrToHex (Nav_pack3_Cor2,Cor32);
        StrToHex (Nav_pack3_Tim21,Tim321);
        StrToHex (Nav_pack3_Tim22,Tim322);

        StrToHex (Temp_1,Temp1);
        StrToHex (Temp_2,Temp2);
        StrToHex (Press_1,Press1);
        StrToHex (Press_2,Press2);


(* for checking purposes

        Writeln(nav_pack1_cor1,'  ', nav_pack1_Tim11,'  ',Nav_pack1_Tim12,
                Cor11:4:0,Tim111:5:0,Tim112:5:0);
        Writeln(nav_pack1_cor2,    , nav_pack1_Tim21,'  ',Nav_pack1_Tim22,

                Cor12:4:0,Tim121:5:0,Tim122:5:0);

        Writeln(nav_pack2_cor1,'  ', nav_pack2_Tim11,'  ',Nav_pack2_Tim12,
                Cor21:4:0,Tim211:5:0,Tim212:5:0);
        Writeln(nav_pack2_cor2,    , nav_pack2_Tim21,'  ',Nav_pack2_Tim22,
                Cor22:4:0,Tim221:5:0,Tim222:5:0);

        Writeln(nav_pack3_cor1,'  ', nav_pack3_Tim11,'  ',Nav_pack3_Tim12,
                Cor31:4:0,Tim311:5:0,Tim312:5:0);
        Writeln(nav_pack3_cor2,    , nav_pack3_Tim21,'  ',Nav_pack3_Tim22,
                Cor32:4:0,Tim321:5:0,Tim322:5:0);
```

```
        Writeln(Temp_1,'  ',Temp_2,'  ',Temp1:5:0,Temp2:5:0);
        Writeln(Press_1,'  ',Press_2,'  ',Press1:5:0,Press2:5:0);
        WriteLn;WriteLn;
*)
        Decimal_Bytes_to_REAL( Cor11 , Tim111 , Tim112 ,
                               Cor12 , Tim121 , Tim122 ,
                               Cor21 , Tim211 , Tim212 ,
                               Cor22 , Tim221 , Tim222 ,
                               Cor31 , Tim311 , Tim312 ,
                               Cor32 , Tim321 , Tim322 ,
                               Temp1 , Temp2  ,
                               Press1 , Press2  );
      End;
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   PROCEDURE Read_Header;

      Begin

        Writeln( Buffer );
(*        IF OutPutFile_Exist THEN
            WriteLn( OutPutFile,Buffer);
*)
      End;
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   PROCEDURE Read_InPutFile ; | InPutFile Lesen

      VAR
         InPut_Buffer : Array[0..80] of Byte ;
         BufferStr    : STRING[80] absolute InPut_Buffer;
         TestDummy : STRING[2];
         Check     : STRING[1];
         Test      : REAL ;
         Base_Error : BOOLEAN ;

  Begin

    REPEAT                    (*    header-info lesen *)
        ReadLn(InPutFile,Buffer);
        Check := Copy(Buffer,39,39); (* 01.03.90 , Abfrage fur neue ROM's *)
        Read_Header;
    UNTIL (Check = ':') ;

    ReadLn(InputFile);        (* Leerzeile im InPutFile uberspringen *)

       WriteLn;                                    (*  Tabelle beschriften *)
       Write('        NavPacket_1        NavPacket_2    ');
       Write('        NavPacket_3            Temp   Press  ');
       Write('   [sec.]     [sec.]       [sec.]     [sec.]  ');
       Writeln('      [sec.]      [sec.]  [',CHAR(248),'C]  [dbar]');
```

```
(*      IF OutPutFile_Exist THEN
        Begin
         WriteLn;
         Write(OutPutFile,'          NavPacket13            Temp  Press ');
         Write(OutPutFile,'  [sec.]     [sec.]      [sec.]      [sec.]  ');
         Writeln(OutPutFile,'      [sec.]      [sec.]  [',CHAR(248),'C] [dbar]');
        End;
*)
      REPEAT        (* Hex-Strings der Daten einlesen *)
          ReadLn(InPutFile,Buffer);
          String_2_Values;
      UNTIL EOF (InPutFile);
      WriteLn;WriteLn(' End of InPutFile ! ');
  End;
 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

    PROCEDURE Oeffnen_InPutFile ; | InPutFile oeffnen

    VAR   InPutName  :   STRING[30];
          exist      :   BOOLEAN;
          i          :   INTEGER;

    BEGIN
      WriteLn;WriteLn;WriteLn;
        REPEAT
           Write(' Name des InPut-Files : ');
           ReadLn(InPutName);
           exist := FileExist(InPutName);
           IF NOT exist THEN
            BEGIN
              SOUND(333); WriteLn;
                Write(' !!!! Achtung , InPut-File > ');
                TextBackground(Red);TextColor(White + Blink);
                Write(InPutName);
                TextBackground(Blue);TextColor(Yellow);
                WriteLn(' < existiert nicht !!!!! ');
              WriteLn;
              WaitKey;
              NOSOUND;
            END
        UNTIL exist ;
          BEGIN
            Assign(InPutFile,InPutName);
            ReSet(InPutFile);
            WriteLn;WriteLn;
            WriteLn('    ****** IntPutFile " ',InPutName,' " geoeffnet ******');
            WriteLn;WriteLn;
          END;
    End;
 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
PROCEDURE Oeffnen_OutPutFile ; | OutPutFile oeffnen bzw. neu anlegen  Ü

VAR   Name    :   STRING[30];
      exist   :   BOOLEAN;
      i       :   INTEGER;

BEGIN
  WriteLn;WriteLn;WriteLn;
    REPEAT
       WriteLn;WriteLn;
       Write(' Name des OutPut-Files : ');
       ReadLn(Name);
       exist := FileExist(Name);
       IF exist THEN
        BEGIN
            SOUND(222); WriteLn;
            Write('!!!! Achtung , OutPut-File > ');
            TextBackground(Red);TextColor(White + Blink);
            Write(Name);
            TextBackground(Blue);TextColor(Yellow);
            WriteLn(' < existiert schon !!!!!');
            WriteLn;
            IF JaNein('      Soll OutPutFile $berschrieben werden (J/N)  ')
THEN
              Begin
                 Assign(OutPutFile,Name);
                 ERASE(OutPutFile);
                 exist := false;
              End;
          WriteLn;
          NOSOUND;
       END
    UNTIL NOT exist ;
       Assign(OutPutFile,Name);
       ReWrite(OutPutFile);
       OutPutFile_Exist := TRUE ;
       WriteLn;WriteLn;
       WriteLn('   ****** OutPutFile  >> ',Name,' << ******');
       WriteLn;WriteLn;
    End;

Procedure SetUp ;

VAR
     Hour , Min ,
     Second , Sec100 ,
     Year , Month ,
     Day , DayOfWeek  : Word;
```

```
  Begin
    TextBackground(Blue);
    TextColor(Yellow);
    ClrScr;
    WriteLn;WriteLn;WriteLn;
    GetDate(  year,month,day,dayofweek );
    GetTime( Hour,Min,Second,Sec100);
    Write('     DECODRAM running on ',Day,'.',Month,'.',year);
    WriteLn('  at  ',Hour,':',Min,':',Second);WriteLn;
    OutPutFile_Exist := FALSE;
    IF JaNein('Sollen Daten auf Disk geschrieben werden (J/N)') THEN
          Oeffnen_OutPutFile;
 End;

BEGIN
   SetUp;
   Oeffnen_InPutFile;
   Read_InPutFile;
   Close(InPutFile);
   IF OutPutFile_Exist THEN
     BEGIN
        WriteLn(OutPutFile,' End of InPutFile !');
        Close(OutPutFile);
     End;
 END.    (* of main code *)I
```

Appendix A3

List of suppliers of major components for RAFOS floats and for sound source moorings in Kiel

Status: Autumn, 1991

No.	Item	Description	Manufacturer/Distributer
RAFOS floats			
1	Boxes, wood	Packing material	Freese, Bordesholm
2	Cast, release plugs alternative	Conathane EN-7 Epoxid LN/LN-H, 4:1	Conap, Inc., Orelan, NY, USA VOSS Chemie, Uetersen
3	Electronic boards	#1203, 1276 (set)	Bathy System, Inc., West Kingston, RI, USA
4	Endplate	Custom designed	Hydrobios, Kiel
5	ERPROMs	27 C 64 O	Enatechnik, Quickborn
6	Glass tube	DURAN (TM)	Schröder, Ellerau
7	Hydrophone	RAFOS type 7881-1	Benthos, N Falmouth, MA, USA
8	ID number	-	CLS, Toulouse, France
9	Magnets	external, for Reed switches	SIS, Kiel
10	Misc. electronics	-	Mütron, München
11	O-rings	for release etc.	Mordhorst und Bockendahl, Kiel
12	Pressure	EAF 2 K with 3/8"- 24 UNK threaded port	Data Instruments, Schöllkrippen
13	Release wire	INCONEL 625	Astrolite, Camerillo, CA, USA
14	Silicon glue	RTV, Sistra F106	Mordhorst und Bockendahl, Kiel
15	Thermistors	Y.S.I. 44032	Delta, München
16	Transmitter	T-2023 II	SIS, Kiel or Toyo Communication Equipment Co., Tokyo, Japan
17	VARTA batteries	Mono (D), Baby (C), Mignon (AA)-40XX	Montanus, Kiel

No.	Item	Description	Manufacturer/Distributer
Sound sources moorings			
1	Sound source	derived from SOFAR floats	WRC, Falmouth, MA, USA
2	Robe	Meteor-Leine	Seldis, Hamburg
3	Glass ball	buoyancy 17'Ø	Benthos N Falmouth, MA, USA
4	Swivel	above and beneath sound source	NSW, Nordenham
5	SW transmitter	on top buoy 27 MHz (CB)	IBAK, Kiel
6	Ac release	RT 161 or equivalent	Oceano Instruments (Mors), Massy, France
7	Parachute	reduction of sinking speed	Zelt-Haase, Kiel
8	Anchor	railway wheel ~300 kg in water	Deutsche Bundesbahn Hamburg-Harburg

Appendix A4

FORTH glossery

contributed by

E. Carter

Electra Software and Consulting

PO Box 51034

Pacific Grove

CA 93950

USA

RAFOS FORTH Glossary

RAFOS Forth V2.2 uses a direct threaded interpreter and is based upon the 1979 Forth model. The following is list of the words that are unique to RAFOS Forth. The stack diagrams use the following notation.

Stack Notation

Symbol	Meaning
n, n1,...	16-bit single-precision integers
d, d1,...	32-bit double-length integers
u	16-bit unsigned single number
ud	32-bit unsigned double-length number
char or c	7-bit ASCII character
byte or b	An 8-bit byte
flag or f	A Boolean flag, a -1 or 0
addr, addr1,...	16-bit Addresses
$ or $addr	The address where a string is to be found

A TRUE flag is a -1 (hexadecimal FFFF), FALSE is zero. Words not listed here behave according to the FORTH 79 model. Port A is at $0000, Port B is at $0001, Port C is at $9003. Port D is on the PRU chip at $9004, it is an output only port and is not used by any part of the present float software.

The interrupt vectors are vectored to jumps at:

$0010	Warm Timer	Interrupt
$0013	Timer	Interrupt
$0016	External	Interrupt
$01D2	Software	Interrupt

A new interrupt is installed by placing the vector to the code at the appropriate address above plus one.

The complete vocabulary listing for version 2.2. The first group of words are RAFOS float specific, the second group are Forth-79. This list was generated by typing the command: VLIST, and then separating the two groups of words.

```
     MISSIOx  NAVIGAxx RR  AON  PP  VV  FINISH  DLY  A_OFF
TELEMExxx  FLASHEx  USE  NO  RAD_STxxx  MSG  CHECK  RANDY
DUPSM  NM  SUMUP  RADR  NMSG VECTOR  SWITCH  TRNSMT  BUFFER
1P!  ZERO?  W160MS  WARM_Ux  DELAY  ID_HASx  ID  TR_OFF
CARR_Ox  OSC_ON  RINIT  RAD_ISx  CYCLE  PT_REAx  TIMEOUx
DOWN  UP  WAIT5  SENSE  INSTRUxx  OFFSET  SYNCHRxxxxx
DEFAULx SKED  Z  INTERVxx  LTT  CORSWAx  WINDOWx  LLEN
INDEX  VPTR  BOTTOM  VFLAG  BIGTIMx  BIGCOR  2TIME  2COR
1TIME  1COR  VDATA  DATA  RSUM TCOR  LTP  TIME?  TOD  SAMPLE
SYNC  PSYNC  SYNCFLxx  PAUSE  CBARS CDEGRExx  TEMP  PRESSUxx
RECV_Ixxx  PT_INIx  2MINT  2MIN  10HZ  CLKINIx  POWER  OFF
PR_ON  GMT  LISTEN  ?BATTExx  BEEP  DSALL  START LOFF  LON
PR_OFF  TOFF  WTISR  PTISR  WISR  TISR  CISR  TINIT  INIT
S>>  >>  <<  SWI  STOP  WAIT  SEI  CLI  CSPLIT

COLD  ?TERMIxxx FORGET  FENCE  VLIST  OUT  C/L  (  ID.  ?
2*  MAX  MIN  SPACE  ABORT MOD  /  /MOD  */  */MOD  M*  M/
D+-  KEY  ."  U.  LEAVE  DECIMAx  HEX  I  DNEGATx  +LOOP
LOOP  DO  <+LOOPx  <LOOP>  <DO>  'STREAx >IN  TIB  <.">
REPEAT  WHILE  ELSE  THEN  IF  UNTIL  AGAIN  BEGIN [COMPIxxx
*  CONSTAxx  VARIABxx  '  :  ;  COMPILx  .  #S  DDUP XOR
AND  OR  #  +-  NEGATE  SIGN  =  >  <  0=  0<  ABS  SMUDGE
BASE  M/MOD  HOLD  ROT  R@  R>  >R  #>  OVER  <#  PAD  S->D
U/MOD U*  -  +  DEFINIxxxxx  ]  [  2+  2  1  0  COUNT  -FIND
IMMEDIxxx  TOGGLE  QUIT  CREATE  CR  SP!  SWAP  LIT  ALLOT
LATEST  +!  DUP C,  ,  C!  !  FORTH  CURRENx  CONTEXx  STATE
HLD  1+  NOT  HERE DP  @  C@  DROP  <NUMBExx  WORD  BL  EMIT
EXECUTx  EXIT  0BRANCx BRAN  <FIND>  TYPE
```

Note that since only the character count and the first six characters are kept in the header, characters beyond the sixth are unknown and are indicated above by an x where the character would be.

The word BASE is a one byte variable (requiring the use of C! and C@) instead of the Forth 79 standard two byte variable. The word NOT is a bitwise operation (as in the Forth 83 standard) not a synonym for 0= (as it is in the Forth 79 standard).

>> (u1 u2 -- u3)

Unsigned Right bit shift of u1 by u2 bits. The high order bits are filled with zeros (Constrast with S>>). Only the lower 4 bits of u2 are used (the other bits are masked off).

<< (u1 u2 -- u3)

Left bit shift of u1 by u2 bits. The low order bits are filled with zeros. Only the lower 4 bits of u2 are used (the other bits are masked off).

?BATTERY (-- flag)

Checks the battery sensor at Port B bit 0, and pushes a TRUE flag if the battery is low (the bit is 0).

10HZ (-- n)

Puts a flag on the stack (hex 40) for subsequent setting of the interrupt rate to ten times per second.

1COR (-- addr)

This variable contains the value of the second largest correlation height from the previous navigation window.

1P! (addr --)

Increments the contents of the address on the stack by one. Implemented as:

```
: 1P!   DUP @ 1+ SWAP ! ;
```

could also have been defined as:

```
: 1P!   1 SWAP +! ;
```

1TIME (-- addr)

This variable contains the time of the second largest correlation height from the previous navigation window.

2COR (-- addr)

This is a scratch variable which is used in manipulating the two largest correlation values in a navigation window. It is only used in the navigation interrupt routine, not from Forth (except to initialize it to zero).

2MIN (-- n)

Puts a flag on the stack (0) for subsequent setting of the interrupt rate to once every two minutes.

2MINT (--)

Starts up the two minute interrupts. These are the "clock ticks" for the time of day clock. Implemented as:

```
: 2MINT   SEI  TISR  17 ! 2MIN PR_ON PR_OFF INIT ;
```

2TIME (-- addr)

This is a scratch variable which is used in manipulating the two largest correlation times in a navigation window. It is only used in the navigation interrupt routine, not from Forth.

A_OFF (--)

Turn off all the bits of Port A. Implemented as:

```
: A_OFF  0  0 C! ;
```

AON (--)

Turn the acoustic receiver board on briefly, for electrical checkout. Implemented as:

```
: AON   10HZ PR_ON DLY PR_OFF ;
```

BEEP (--)

Sends an ASCII beep character to the terminal. Implemented as:

```
: BEEP  7 EMIT ;
```

BOTTOM (-- addr)

A variable that contains the greatest pressure (in centibars) that the mission code will allow before deciding to abort the mission. Initialized by DEFAULT to 12000 centibars.

BIGCOR (-- addr)

A variable that contains the largest correlation height found during the previous navigation window.

BIGTIME (-- addr)

A variable that contains the time (in tenths of seconds since the window start) when the highest correlation height (BIGCOR) was found.

BUFFER (-- addr)

This variable points to the start of the radio message data buffer.

CARR_ON (--)

Turns on the radio carrier signal by sending the proper bit mask to the radio transmitter (at Port B). Implemented as:

```
HEX : CARR_ON 3D 1 C! ;
```

CBARS (n1 -- n2)

Converts the frequency count on the stack to hundredths of a Bar pressure.

CDEGREES (n1 -- n2)

Converts the frequency count on the stack to hundredths of a degree Centigrade.

CHECK (-- n)

Calculates the checksum of the radio message buffer. Implementation:

```
: CHECK
    BUFFER 7 + C@
    37 7 DO
        RANDY
        BUFFER I + 1 + C@
        XOR
    LOOP
    RANDY
;
```

CHOOSE (n1 -- n2)

This word chooses a random number that is less than the value given on the stack. I.E. n2 = random value in the range 0 .. n1 - 1. Implemented as:

```
: CHOOSE   RAND SWAP MOD ;
```

CISR (--)

This word is the entry to the correlator interrupt service routine. This word SHOULD NOT BE USED FROM FORTH, it is for interrupts only. It is in the dictionary so that the interrupt vector can be multiplexed with other interrupts depending upon the mission context.

CLI (--)

Clear the interrupt inhibit mask. This enables the processor interrupt system.

CLKINIT (--)

Initialize the software real time clock. Does not start it running.

CORSWAP (--)

Conditionally swaps the correlations around so that after the latest measurement the following is true: BIGCOR/BIGTIME holds the largest correlation found so far and its time. 1COR/1TIME holds the second largest correlation, and its time, that is at least 4 seconds different from BIGTIME.

CSPLIT (n1 -- u2 n3)

This word takes a 16 bit number on the stack and splits the high and low bytes apart. The result is in the low byte of the two values on the stack. The high order byte is sign extended and is on the top of the stack, the low order byte is taken as unsigned.

CYCLE (--)

This word runs the complete Vocha cycle. It makes pressure and temperature measurements at each stage and stores the measurements in VDATA. If everything runs normally, VFLAG is decremented by one at the end of the cycle. This word may end early if a failure of the mechanism is

detected, in which case VFLAG will be negative reflecting that a failure occurred (see VFLAG).

DATA (-- addr)

DATA is the address of the start of the main data storage buffer. It is open ended (i.e. there is no explicit check for reaching the end of the allocated data storage space).

DEFAULT (--)

This word initializes the various mission oriented variables to their default values. It is implemented as:

```
: DEFAULT   DECIMAL  LOFF
            15000 LLEN        !
            270  WINDOWS      !
            12000 BOTTOM      !
            0     VFLAG       !
            30    INTERVAL    !
            0     INDEX       !
            0     VPTR        !
            0030 GMT LTT      !
            0100 GMT LTT  2+  !
            0130 GMT LTT  4 + !
            0830 GMT LTT  6 + !
            0900 GMT LTT  8 + !
            0930 GMT LTT 10 + !
            1630 GMT LTT 12 + !
            1700 GMT LTT 14 + !
            1730 GMT LTT 16 + !
            0    Z
            CLKINIT 2MINT
;
```

DELAY (-- n)

This word waits for the amount of time required by ARGOS between radio transmissions. Leaves a zero on the stack for backward compatibility with an older version that had two exit conditions. Implemented as:

```
: DELAY
    0        ( Push a normal exit flag for backward compatibili
    RAD_START RINIT
    ID_HASH 0 DO
                  W160MS
                  LOOP
    LOOP
    2MINT
;
```

DOWN (--)

This word turns on the Vocha motor in the direction that will move the float downward in the water column. DOWN will pause for 3 to 5 seconds before returning so that the limit switch can settle before being sensed by subsequent words. (See WAIT5).

DLY (--)

A word to generate a short delay for use during the electrical checkout proceedures. Implemented as an empty DO loop:

```
HEX   : DLY   3000 0 DO LOOP ;
```

DSALL (addr --)

This word moves 36 bytes, starting at the given address, bitwise, out to the radio transmitter input which is at bit 7 of Port B. It sends two bits for every data bit: a 10 sequence if the given data bit is 0, or a 01 sequence if the given bit is a 1. The timing of this word is carefully desgined to comply with ARGOS requirements.

DUPSM (*n* -- *n*)

Increments RSUM by one if the value on the stack is even. Part of the checksum calculation. Implementation:

```
: DUPSM   DUP 2 MOD IF RSUM 1P! THEN ;
```

EXCHANGE (n1 n2 --)

Exchanges elements n1 and n2 of the random number array. Implemented as:

```
: EXCHANGE   OVER OVER S@ SWAP S@ ROT S! SWAP S! ;
```

FINISH (--)

This word turns zeros all the bits of Port A after a pause. It pauses before returning as well. Designed as part of the electrical checkout words. Implemented as:

```
: FINISH   DLY A_OFF DLY ;
```

FLASHER (n --)

This word expects a mask on the stack that will control turning the strobe light flasher (at PORT A) on or off. (See USE and NO). Implemented as:

```
: FLASHER    0  C!  ;
```

GMT (n1 -- n2)

Converts the time of day in HHMM format to internal (two minute count) format. Implemented as:

```
: GMT   100 /MOD 60 * + 2 /  ;
```

ID (-- n)

A constant containing the lowest 16 bits of the floats ARGOS ID number.

ID_HASH (-- n)

This word uses the floats ARGOS ID number in order to calculate the radio repetition rate. Implemented as:

```
HEX  : ID_HASH   ID ABS 31415 32767 */ 1+ 160 MOD 450 + ;
```

INDEX (-- addr)

A variable whose contents are the offset from the beginning of the (primary) data buffer (DATA) that the next measurement values will be stored at. In other words the next measurement will be stored at the location:

```
INDEX @ DATA +
```

INIT (--)

This word enables the off-CPU part of the interrupt system. The implementation is equivalent to:

```
HEX   : INIT   SEI 43  9003 C! CLI ;
```

INSTRUCT (--)

This word issues basic instructions about how to start up the float mission. It gives opportunities to break out of the mission if the user decides to do so.

INTERVAL (-- addr)

A variable that contains the number of two minute intervals to wait between stages of the Vocha cycle before going on to the next stage. This variable indirectly controls the length of a Vocha cycle. Initialised by DEFAULT to 30 (i.e. 60 minutes).

LISTEN (--)

This word controls the timing for the listening to the acoustic signal. Since the actually listening is done with an interrupt routine, this routines role is just to allow the interrupt to be serviced. Implemented as:

```
: LISTEN   0 SYNCFLAG C!   CLI ;
```

LLEN (-- addr)

A variable that contains the length of a navigation window (in tenths of seconds). Controls the amount of time that the float will listen for the signal from the sound sources. Initialised by DEFAULT to 15000 deciseconds (25 minutes).

LOFF (--)

Turns off the release circuit and its LED by writing a zero to bit 1 on Port B (at $0001).

LON (--)

Turns the release circuit and its LED on by writing a one to bit 1 on Port B (at $0001).

LTP (-- addr)

LTP is the Listening Time Pointer variable, that indexes into the appropriate place in the Listening Time Table (LTT). When properly set it points to the next acoustic listening time. LTP is a variable, it must be dereferenced to get the offset from the start of LTT.

LTT (-- addr)

LTT is the Listening Time Table. The value left on the stack is the address of the first entry in the table. It is implemented equivalently to:

```
VARIABLE LTT  16 ALLOT
```

It is initialized by DEFAULT as follows:

```
0030 GMT LTT      !
0100 GMT LTT  2+  !
0130 GMT LTT  4 + !
0830 GMT LTT  6 + !
0900 GMT LTT  8 + !
0930 GMT LTT 10 + !
1630 GMT LTT 12 + !
1700 GMT LTT 14 + !
1730 GMT LTT 16 + !
```

MISSION (--)

Runs the default sampling mission. Implemented as:

```
: MISSION

    CR
    HEX ." Float ID: " ID U.   ( . is used in earlier ROMS )
    DECIMAL ." The time is: "  TIME? CR

    LON
    INSTRUCT

    ( Wait for first interrupt or keyboard )
    TOD @     ( Put TOD on Stack )
    1+        ( increment it )
    DUP 720 = IF DROP 0 THEN     ( reset if midnight )

    BEGIN ?TERMINAL IF LOFF EXIT THEN DUP TOD @ = UNTIL

    DROP

    LOFF      ( Turn off  the release light )

    BEEP

    ." OK you can unplug me now. " CR


    LSTMSG      ( display final message )

    0 DATA C!    ( store status byte 0 = OK )

    2 INDEX !   1 VPTR !

    ( Initilize the LTP )
    0 8 0 DO TOD @ LTT I 2* + @ < IF DROP I 2* LEAVE THEN LOOP
    LTP C!

    WINDOWS @ 0 DO
                I .
                ( WAIT FOR LISTENING TIME )
                SKED
```

```
                TIME?

                LTP C@ 2+ LTP C!

                CLKINIT

                NAVIGATE

                ( Now take a pressure & temperature measurement )
                LTP C@ 6 MOD 0=
                IF
                   INDEX @ DATA +
                   PT_READ
                   INDEX @ 4 + INDEX !  ( bump pointer )

                   ( The Vocha cycle goes here )
                   VFLAG @ IF CYCLE THEN

                THEN

                ?BATTERY IF 66 DATA C! ( status = 'B' ) LEAVE THEN
                PRESSURE SAMPLE CBARS BOTTOM @ >
                         IF 80 DATA C! ( status = 'P' ) LEAVE THEN

        LOOP   ( to next listening window )

        2MINT

        LON    ( Turn on the release light )

         ( WAIT A HALF HOUR )
         15  SYNCHRONIZE

         LOFF

         ( Calculate the number of messages and store that )
         NM NMSG @ 1 DATA + C!
         ( Store VOCHA status )
         VFLAG @ VDATA C!

        BEEP ." Coming Home ! " CR

         ( start radio message on the half hour )
         BEGIN TOD @ 15 MOD WHILE STOP REPEAT
         TELEMETRY
```

The final message is a headerless word (i.e. it does not appear in the Forth dictionary) that is equivalent to: ." Window TOD Time Cor " CR

MSG (n --)

Builds up the message that should be sent out to the radio transmitter next. Implemented as:

```
: MSG
    BUFFER 7 + C!    ( store message number )
    31 0 DO
         I RADR @ + VECTOR @ + C@ ( get to the next byte )
         BUFFER 8 + I + C!        ( store it where it belongs )
    LOOP
    CHECK            ( do the checksum )
    BUFFER 6 + C! ( and store it )
    31 RADR +!       ( update RADR )
;
```

NAVIGATE (--)

This word runs the correlator in a busy-wait loop in order to aquire the navigation data. It saves the two largest correlation values (divided by two) and the offset into the navigation window (in tenths of seconds) that the correlation values were found at. Implemented as:

```
: NAVIGATE
        0 BIGCOR !   0 1COR !  0 2COR !  0 1TIME !

        ( NOW SET 10Hz CORRELATION RATE )
        RECV_INIT

        LLEN @ 0 DO
                LISTEN
                SYNC
                RSUM @ DUP
                BIGCOR @ > IF CORSWAP
                                BIGCOR ! TCOR @ BIGTIME !
                            ELSE DROP THEN
        LOOP

        PR_OFF

        2MINT

        TIME?

        BIGTIME ? BIGCOR ? CR

        ( at this point save the correlation data  )

        INDEX @ DATA +
        BIGCOR  @ 2 / OVER C! 1+
        BIGTIME @ OVER ! 2+
        1COR    @ 2 / OVER C! 1+
        1TIME   @ SWAP !

        INDEX @ 6 + INDEX !     ( bump data pointer )
```

```
;
```

NM (--)

Determines the number of radio messages the stored data will need and stores the result in NMSG, also stores the message boundary between the normal data and the Vocha data in SWITCH. Implementation:

```
: NM
    INDEX @ 31 /MOD SWAP IF 1+ THEN
    DUP
    SWITCH !
    VPTR @ 31 /MOD SWAP IF 1+ THEN
    +
    NMSG !
;
```

NMSG (-- addr)

This variable contains the total number of radio messages that will be required to send the complete data set (both the main and the Vocha data buffers).

NO (-- n)

A bit mask, leave a zero on the stack so that a subsequent invocation of FLASHER will turn off the strobe flasher.

OFF (--)

This word turns the acoustic receiver board off, by writing a zero to Port C. Implemented as:

```
: OFF    ZERO  PR_ON ;
```

OFFSET (--)

This interactive word is run in order to determine the offset of the internal real time (two minute) clock.

OSC_ON (--)

Turns on the radio oscillator by setting the proper bits of Port B. Implemented as:

```
HEX    : OSC_ON    1D  1 C! ;
```

PAUSE (--)

A word for pausing for a short interval, for controlling timing. Implemented as an empty DO loop:

```
HEX   : PAUSE   20 0 DO LOOP ;
```

POWER (n --)

POWER is used to turn on the acoustic receiver board. It takes the lower 8 bits of the value on the stack as a control mask, adds the control bit for power-up and sends it to Port C. Implemented as:

```
HEX    : POWER   60 OR 9003 C! ;
```

PP (--)

This word takes a temperature and pressure measurements and displays the readings. Designed for the electrical checkout procedure. Implemented as:

```
: PP
    BASE C@  DECIMAL          ( Save the base and set to decimal )
    TEMP SAMPLE CDEG .
    PRESSURE SAMPLE CBARS .
    CR
    BASE C!                   ( Restore the base )
;
```

PR_OFF (--)

Turns off the interrupts coming from the PRU chip. Impelemented equivalently to:

```
: PR_OFF   Port_C C@ Port_C 3 + C@ DROP DROP 3 Port_C C! ;
```

PR_ON (n --)

This word sends the lower 8 bits of the mask on the stack to Port C for controlling the acoustic receiver board. Typical usage:

```
ZERO PR_ON  ( turn off the receiver board )
```

This word is implemented as:

```
HEX     : PR_ON    9003   C!   ;
```

PRESSURE (-- n)

A bit mask, leaves a 16 on the stack so that a subsequent invocation of SAMPLE will read the pressure sensor.

PSYNC (n --)

This word causes the CPU to pause until the synchoronization flag (SYNCFLAG) has been incremented to the value specified on the stack. Normally used to wait a specified number of tenths of seconds before proceeding on. Implemented as:

```
: PSYNC   0 SYNCFLAG C! CLI
          BEGIN WAIT DUP SYNCFLAG C@ = UNTIL
          SEI DROP
;
```

PT_INIT (--)

Initialize the pressure/temperature measurement system. This includes setting up the interrupts for the measurement, leaving them running at the ten Hertz rate. Implemented as:

```
: PT_INIT   SEI PTISR 17 ! WTISR 14 ! 10HZ PR_ON PR_OFF INIT ;
```

PT_READ (addr --)

Reads the temperature and pressure sensors, and stores the results in two sequential locations starting at the address given on the stack. Implemented as:

```
: PR_READ
      TEMP SAMPLE      ( Get the temperature )
      CDEGREES         ( Convert to centidegrees )
      OVER !           ( store it )
      2+               ( calculate next storage addr )
      PRESSURE SAMPLE  ( Get the pressure )
      CBARS            ( Convert to centibars )
      SWAP !           ( store it )
;
```

PTISR (--)

This word is the entry to the timer interrupt service routine for use during pressure and temperature measurements. This word SHOULD NOT BE USED FROM FORTH, it is for interrupts only. It is in the dictionary so that the interrupt vector can be multiplexed with other interrupts depending upon the mission context.

RAD_ISR (--)

This word is the entry to the radio transmitter interrupt service routine for use during the telemetry of the aquired data. This word SHOULD NOT BE USED FROM FORTH, it is for interrupts only. It is in the dictionary so that the interrupt vector can be multiplexed with other interrupts depending upon the mission context.

RAD_START (--)

Takes the actions necessary to start up the radio. This includes setting the interrupt vectors, initializing the radio data buffer, and determining the number of messages that need to be sent. Implemented as:

```
HEX     : RAD_START
            SEI
            [ ' RAD_ISR ] DUP 11 ! 14 !  ( set the timer vectors )
            CLI
            ( Initialize the data buffer )
            FFFE BUFFER !  2FFA BUFFER 2+ !  ID BUFFER 4 + !
            0 RADR !
            DATA VECTOR !
            NM          ( determine the number of messages )
          ;
```

RADR (-- addr)

A variable that contains the location of the last byte that was sent to the radio data buffer.

RAMP (--)

Generates a sequential sequence of numbers in the random number array. Implemented as:

```
: RAMP   NMSG @ 0 DO I I S! LOOP ;
```

RAND (-- n)

This word returns a random integer. Implemented as:

```
: RAND   SEED @ 31415 32767 */ 1+ DUP SEED ! ;
```

RANDY (n -- n)

Part of the checksum calculation routine. Implementation:

```
: RANDY
     0 RSUM !
     ZERO? DUP
     DUPSM 2 >>
     DUPSM 1 >>
     DUPSM 1 >>
     DROP SUMUP
;
```

RECV_INIT (--)

Initialize and power up the acoustic receiver in preparation for aquiring acoustic travel time data for navigation. Implemented as:

```
: RECV_INIT   SEI CISR 17 ! 10HZ PR_ON PR_OFF INIT ;
```

RINIT (--)

This word initializes the timer port appropriate to starting up the radio.

RR (--)

This word builds up a dummy data buffer and sends it out the radio transmitter for a system checkout of the radio. Implemented as:

```
: RR
    36 6 DO 0 I BUFFER + C! LOOP  ( zero navigation data
    TEMP SAMPLE CDEG BUFFER 6 + ! ( Temperature data )
    PRESS SAMPLE CBARS BUFFER 10 + ! ( Pressure data )
    RAD START
    TRANSMIT
;
```

RSUM (-- addr)

This word is a variable that contains the real part of the complex correlation when the correlator is running, it holds the magnitude of the complex vector when the correlator has finished.

S>> (n1 u2 -- n3)

Signed Right bit shift of n1 by u2 bits. The high order bits are sign extended (that is they filled with zeros if n1 is positive, and filled with ones if n1 is negative; constrast with >>). Only the lower 4 bits of u2 are used (the other bits are masked off).

S@ (n1 -- n)

Fetches the desired value (element n1) from the random number array. Implemented as:

```
: S@   PAD + C@ ;
```

S! (n1 n2 --)

Stores the desired value (n1) in the array used for the random number array at element n2. Implemented as:

```
: S!   PAD + C! ;
```

SAMPLE (n1 -- n2)

Samples the Temperature or Pressure according to the value on the stack (0 for Temperature, nonzero for Pressure), and leaves the frequency count on the stack. Example usage:

```
TEMP SAMPLE CDEGREES .
```

SEED (-- addr)

A variable that contains a seed value for the random number generator. Note that the location of this variable is the same as the variable 2COR, since 2COR is no longer needed when SEED is being used.

SEI (--)

Set the interrupt inhibit mask in order to block the servicing of interrupts.

SENSE (-- flag)

Checks the Vocha limit switch at Port A bit 2 and leaves a TRUE flag if the switch is turned off (the bit will be low -- the limit switch is in the Vocha dimple).

SHUFFLE (--)

Shuffles the random number array to a random order. Implemented as:

```
: SHUFFLE   NMSG @ 0 DO NMSG@ CHOOSE I EXCHANGE LOOP ;
```

SKED (--)

This word causes the CPU to wait until the next scheduled listenting window. Implemented as:

```
HEX    : SKED
             2MINT
             BEGIN STOP LTP C@ LTT + @ LTIM @ = UNTIL
       ;
```

START (--)

Starts up the period counter for making a pressure or temperature measurement. This word includes a fail-safe, in case the counter never counts.

STOP (--)

Stop the cpu until an internal interrupt is received. When this word is executed the system stops including the internal timer (compare with WAIT).

SUMUP (n -- n)

Part of the checksum calculation, takes the value on the stack divided by two and combines it with the value stored in RSUM. Implementation:

```
HEX    : SUMUP   1 >> RSUM @ 2 MOD 80 * + ;
```

SWI (--)

When this word is executed a software interrupt is generated which causes the processor to immediately branch to $01D2 where a machine code service routine is expected to be found.

SWITCH (-- addr)

This variable contains the message number of the last radio message that contains the data from the area pointed to by DATA. Radio messages after this value are messages pertaining to the Vocha data stream.

SYNC (--)

The execution of this word causes the CPU to loop until the value of the variable SYNCFLAG is nonzero. For controlling the timing of time dependent operations Implemented as:

```
: SYNC   BEGIN SYNCFLAG C@ UNTIL SEI ;
```

SYNCFLAG (-- addr)

SYNCFLAG is a one byte variable whose value is used in controlling the timing of various time dependent operations. Note that since this is a one byte variable, C! and C@ are used to set and obtain it value.

SYNCHRONIZE (n --)

The execution of this word causes the CPU to hold in place until the two minute time of day counter increments by the amount on the stack. It accounts for crossing midnight. This word is used both for pausing for multiple minutes and to start up various events immediately after the two minute interrupt has occurred. It is implemented as:

```
: SYNCHRONIZE   TOD @ + DUP 719 > IF 720 - THEN
                BEGIN  STOP DUP TOD @ = UNTIL DROP ;
```

TELEMETRY (--)

This is the word that actually transmits the stored data with the ARGOS transmitter. It randomly buffers out the data, alternating between the normal data buffer and the Vocha data buffer. It repeats the data until a keystroke is detected on the input line. The strobe flasher is turned on between transmissions at night time. The messages are sent in a randomly shuffled order so that a partial collection of transmissions has some chance of getting data throughout the data collection mission. The implementation is as follows:

```
: TELEMETRY
      RAD_START
      1234 SEED ! RAMP SHUFFLE ( determine message order )
      BEGIN
        0 RADR !
        DATA VECTOR !          ( Choose the main data buffer )
        0                      ( clear exit flag )
        NMSG @ 0 DO
             ( Check to see if Vocha data buffer is to be sent )
             I S@ SWITCH @ - 0<
                 IF I S@ 30 * RADR ! DATA VECTOR !
                 ELSE I S@ SWITCH @ - 30 * RADR ! VDATA VECTOR !
                 THEN
             NO FLASHER    ( Turn off flasher for transmission )
             I S@ MSG TRNSMT
             ( Check to see if flasher should be turned on )
             TOD @ DUP 2300 GMT > SWAP 1100 GMT < OR
                   IF USE FLASHER THEN

             ( Wait before transmitting again )
             DELAY

             ( conditionally set exit flag )
             IF DROP 1 LEAVE THEN
         LOOP
         ( Check to see if we should quit )
    UNTIL
;
```

Note that since only the first 6 characters are used TELEMETRY can also be referred to as TELEMETER.

TCOR (-- addr)

A variable that contains a temporary value of the correlation height that is obtained while the navigation matched filter correlator is running.

TEMP (-- n)

A bit mask, leaves a zero on the stack so that a subsequent invocation of SAMPLE will read the temperature sensor.

TIME? (--)

Displays the system time in HHMM format.

TIMEOUT (--)

This word stops the Vocha motion either by sensing that the limit switch has reached it desired location or by timing out. The timeout is determined by the occurence of a two minute tick during the time that the motor is still on. It is assumed that the motor is started up immediately after a clock tick through the use of a SYNCHRONIZE call, i.e.:

```
1 SYNCHRONIZE UP TIMEOUT
```

TINIT (--)

Initializes and enables Timer Interrupts. Implemented equivalently to:

```
HEX : TINIT SEI 7 TCR C! FF TIMER C! CLI ;
```

TISR (--)

This word is the entry to the timer interrupt service routine. This word SHOULD NOT BE USED FROM FORTH, it is for interrupts only. It is in the dictionary so that the interrupt vector can be multiplexed with other interrupts depending upon the mission context.

TOD (-- addr)

Stacks the address of the Time-Of-Day counter variable. The value pointed to by this address is in the internal (2 minute count) format.

TOFF (--)

Turns off the timer interrupts. Implemented equivalently to:

```
HEX   : TOFF   47 TCR C! ;
```

TR_OFF (--)

Turns off the radio transmitter by sending the proper bit mask to Port B. Implemented as:

```
HEX   : TR_OFF   0D 1 C! ;
```

TRNSMT (--)

Transmits the previously prepared message buffer with the ARGOS radio. Implemented as:

```
: TRNSMT
    RINIT
    OSC_ON  WARM_UP
    CARR_ON W160MS
    BUFFER  DSALL
    TR_OFF
    TOFF
    2MINT
;
```

UP (--)

This word turns on the Vocha motor in the direction that will move the float upward in the water column. UP will pause for 3 to 5 seconds before returning so that the limit switch can settle before being sensed by subsequent words. (See WAIT5).

USE (-- n)

A bit mask, leaves a two on the stack so that a subsequent invocation of FLASHER will turn on the strobe flasher.

VDATA (-- addr)

VDATA is the address of the start of the data storage buffer for the VOCHA data. It is open ended just like DATA is.

VECTOR (-- addr)

This variable contains the address of the start of the data buffer presently being transmitted (either DATA or VDATA). Notice that this means that VECTOR must be dereferenced twice to actually reach the data.

VFLAG (-- addr)

VFLAG is a variable that controls the use of the VOCHA mechanism. If it is zero, the VOCHA is not used. If it is positive, it indicates the number of VOCHA cycles that remain to be done. If it is negative, it indicates a failure of the mechanism. If it failed, the failure mode can be interpreted in the following way:

```
VFLAG @ ABS EMIT
```

If an 'S' results, then the mechanism stalled. If a 'T' is returned then the mechanism timed out.

VPTR (-- addr)

A variable whose contents are the offset from the beginning of the Vocha data buffer (VDATA) that the next measurement values will be stored at. In other words the next Vocha measurement will be stored at the location:

```
VPTR @ VDATA +
```

VV (--)

This word exercises the relay board (on Port A) that controls the Vocha mechanism, it was designed as part of the electrical checkout procedure. Implemented as:

```
HEX   : VV
          1 0 C! FINISH
         10 0 C! FINISH
          8 0 C! FINISH
          2 0 C! FINISH
         1B 0 C! DLY A_OFF
        ;
```

W160MS (--)

When this word executes, the CPU pauses for 160 milliseconds. This word is used to control the timing of sending data bits to the radio transmitter.

WAIT (--)

Enter a cpu WAIT condition. The processor will stop until an interrupt is received when this word is executed. The internal timer will still run. (Compare with STOP).

WAIT5 (--)

Causes the CPU to mark time for 5 seconds, by using a call to PSYNC. Note: in some ROMS a there is also a word WAIT3, for waiting for 3 seconds. These two were both used to experiment with finding an optimal debounce period for the Vocha limit switch (see UP and DOWN).

WARM_UP (--)

This word causes the CPU to mark time long enough for the ARGOS radio transmitter to properly warm-up. Implemented as:

```
HEX : WARM_UP 400 0 DO WAIT LOOP ;
```

WINDOWS (-- addr)

A variable that contains the number of navigation windows for the mission. This variable indirectly controls the length of the deployment of the float. Initialized by DEFAULT to 270 windows (at nine windows per day, this is thirty days).

WISR (--)

This word is the entry to the interrupt from wait state service routine. This word SHOULD NOT BE USED FROM FORTH, it is for interrupts only. It is in the dictionary so that the interrupt vector can be multiplexed with other interrupts depending upon the mission context.

WTISR (--)

This is the entry to the interrupt for the watchdog timer interrupt service routine. The interrupt is used in the pressure/temperature measurement routine to provide a time-out exit in case a sensor fails (this prevents the float from waiting forever for a signal that never can arrive). This word SHOULD NOT BE USED FROM FORTH, it is for interrupts only. It is in the dictionary so that the interrupt vector can be multiplexed with other interrupts depending upon the mission context.

Z (n --)

Takes the number on the stack as the time of day in HHMM format, converts it to internal format and stores the result in TOD. This word also sets the listening time pointer, LTP, to point to the appropriate location in the listening time table, LTT.

ZERO? (n -- n)

Tests the value on the stack for zero, leaves a HEX FF if it is nonzero, a 0 if it is zero. Implemented as:

```
HEX : ZERO?  DUP 0= IF DROP DROP FF THEN ;
```