

DELFT UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering
Telecommunication and Traffic
Control Systems Group

Analysis of the Eurofix Data Link

A.M. Noorbergen

Date: June 1993

Contents: In the Eurofix concept, a data link for transmission of Differential GPS data is constituted by additional phase modulation of Loran-C signals. This report contains an analysis of the properties of this datalink.

Professor: D. van Willigen
Mentors: M. Braasch, E.J. Breeuwer, D.J.R. van Nee
Assignmentnumber: A-458
Period: August 1992 - June 1993

After the introduction of the Eurofix concept for transmission of Differential GPS data via Loran-C signals by additional phase modulation, the consequences of this modulation for conventional (non-Eurofix) users is discussed. In what follows, various modulation schemes will be introduced and the properties of these schemes will be described.

Indexing terms: Radio Navigation, Phase Modulation, Coding, Radio Propagation,
Integrated Navigation, Loran-C, DGPS, Eurofix

Summary

Eurofix is a combination of Differential GPS and Loran-C, in which DGPS corrections are transmitted to a user via additional phase modulation of the Loran-C signal. The nature of the Loran-C signal structure only allows for very small bit rates. Despite these small bit rates, position accuracies of 8 to 20 meter in the horizontal plane (95 %) can be achieved by utilising atomic frequency references that are already present in Loran-C transmitters. The specifications, set in the Federal Radio navigation Plan for Harbour Approaches (1992) can be met. Using Eurofix, the user has a low-cost, accurate positioning system at his disposal with an improved integrity and availability, compared to standard GPS or standard Loran-C.

The phase modulation mentioned above consists of advancing and delaying individual Loran-C bursts alternately with an amount in the order of 1 μ s. By using balanced coding, the modulation will hardly be noticed by a conventional receiver. A slightly modified Loran-C receiver will be able to detect those individual burst shifts and will be able to decode the DGPS data.

Various modulation methods are feasible. The performance (data rate and bit-error probability) of each of these methods is discussed. This facilitates choosing an optimum in the trade-offs between data rate and bit-error probability, between number of error control bits and effective and between data link performance and "nuisance" for the conventional Loran-C user.

Table of contents

1	Introduction	5
2.	Introduction to Eurofix	7
2.1	Differential GPS	7
2.2	The Eurofix solution	7
2.3	The Eurofix datalink	8
2.4	Why Eurofix?	9
3.	Eurofix: Painfull for conventional Loran-C users?	11
3.1	Signal-to-Noise Ratio in Eurofix	11
3.2	Zero-crossings of the Eurofix signal	12
3.3	Early-late ratios of Eurofix modulated Loran-C signals	14
3.4	Minimization of Eurofix nuisance for conventional Loran-C users	14
3.5	Conclusion: Eurofix is not painfull for conventional Loran-C users	15
4.	Eurofix: data link valuable for DGPS users?	17
4.1	Introduction	17
4.2	The Eurofix DGPS service area	18
4.3	Skywave influences	20
4.4	Forward Error Correction	22
4.5	Perfomance analysis of various modulation schemes	23
4.6	Conclusion: data link is valuable for DGPS users	29
5.	Conclusions and Recommendations	31
	Literature references	33
	APPENDIX A: Tracking offsets and early-late ratios in Eurofix	35
	APPENDIX B: Calculation of Bit-Error-Probabilities in Eurofix	39
	APPENDIX C: Sampled data values for various skywave conditions	45

1. Introduction

Integrated navigation is a topic of great interest at the moment and the years to come. International borders disappear at a quick rate and people have become more mobile. This generates larger flows of traffic on the already overcrowded infrastructures. The growing traffic load can be fed into channels by utilising the achievements of modern technology in the field of radio navigation systems.

Safety and integrity play an important role in the development of (new) radio navigation systems: people entrust their lives to those systems. That is why demands on radio navigation systems have become more stringent. Authorities oblige large vessels for instance to have at least two independent navigation systems aboard. Integration of radio navigation systems opens ways to a larger accuracy and a larger integrity by combining the strong points of 2 or more systems.

At the Delft University of Technology, department of Electrical Engineering, research is done on the integration of Loran-C and GPS under the name EUROFIX. In the Eurofix proposal the Loran-C system is not only used for navigation, but also for data transmission. This data transmission feature will be used to transmit DGPS data from a DGPS reference station to a mobile user. With a slightly modified Loran-C receiver, the user will have DGPS information at his disposal, which offers him a greater accuracy (10 to 20 meter) and a greater integrity. In this way, DGPS corrections are available in a large part of Europe at minimal user cost.

A preliminary Eurofix system has already been proposed at several occasions. But because there are a lot of new transmitters planned in Europe, it is important to convince the authorities to consider to implement Eurofix in these new transmitters. For that reason some haste

was imperative, and just for that reason it was decided to create a Eurofix test set-up. One part of my graduation work was to modify a Dynamic Dual-Chain Loran-C simulator, to make it possible to transmit Eurofix data in its preliminary form [11]. Together with a modified receiver it is then possible to show that Eurofix can work basically.

The datalink that is constituted in the Eurofix proposal hasn't been the subject of a thorough investigation. It is merely based on the ideas of the Clarinet Pilgrim system of the US Navy, in which the 6 last bursts of a group of eight of a station are somewhat shifted with respect to each other. Using different shift patterns, data can be transmitted. The other part of my graduation work was to investigate this data link and to find an answer to the following questions: can the modulation principle be improved, what data-rates can be achieved, what bit-error rates can we expect and what can we do about bit errors, and how will Eurofix affect conventional users?

The report in front of you describes the second part of my graduation work: the analysis of the properties of the Eurofix data link.

2. Introduction to Eurofix

As mentioned in the introduction, Eurofix is a combination of Loran-C and GPS. Why this combination offers additional value to a user, and how this is done, will be explained in this section.

2.1 Differential GPS

The Global Positioning System (GPS) is a satellite-based radio navigation and positioning system, that is being operated by the United States' Department of Defence. This system offers authorised users accuracies of 10 to 20 meter (95 percentile level). Civilian users will have access to a less accurate system: 100 meters. This worse accuracy is due to, among others, the deliberate disturbances, called Selective Availability, that the US Department of Defence introduces into the "civilian signal".

However, these inaccuracies can be diminished for a great deal with Differential GPS: a GPS receiver at an exactly known location can calculate the error that is introduced by S.A. When a mobile user compensates his own GPS-position for this error, he will have a better accuracy. The ultimate mobile user DGPS accuracy depends on the age of the compensation message, because S.A. varies in time and on distance to the reference station (because the common S.A. error decorrelates in space). The only problem is: how to supply a mobile user this correction information? For this, a data communication link is needed. The International Association of Lighthouse Authorities has proposed to let radio beacons transmit DGPS data. In Eurofix, Loran-C can constitute such a data link. Although nothing has been decided yet on the physical data link, the data that should be transmitted is already standardised by the Radio Technical Commission for

Maritime Services (RTCM). This DGPS standard is known as SC-104.

2.2 The Eurofix solution

The Eurofix concept integrates Navstar/GPS with Loran-C. Differential GPS corrections are phase coded onto the Loran-C signals. This can be done quite easily and cheaply. The data rates that can be established vary from 10 to 25 bits per second, however, Beekhuis [10] showed that, even with such a low data rate, a very good DGPS performance can be achieved (better than the specifications of the Federal Radio navigation Plan for Harbour Approaches), at the cost of using atomic references in the DGPS reference station.

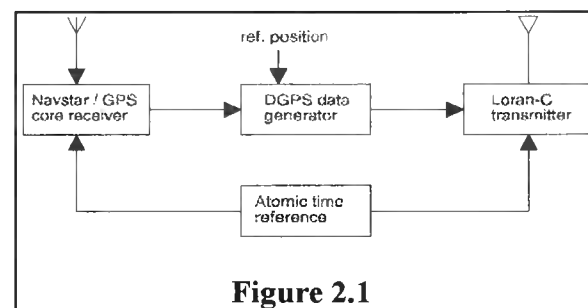


Figure 2.1

From a safety point of view, a lot of might want to have both a GPS- and a Loran-C receiver at their disposal. They only need a slightly modified Loran-C receiver to have Differential GPS corrections at their disposal as well. The Eurofix receiver concept can be described on the basis of the following figure:

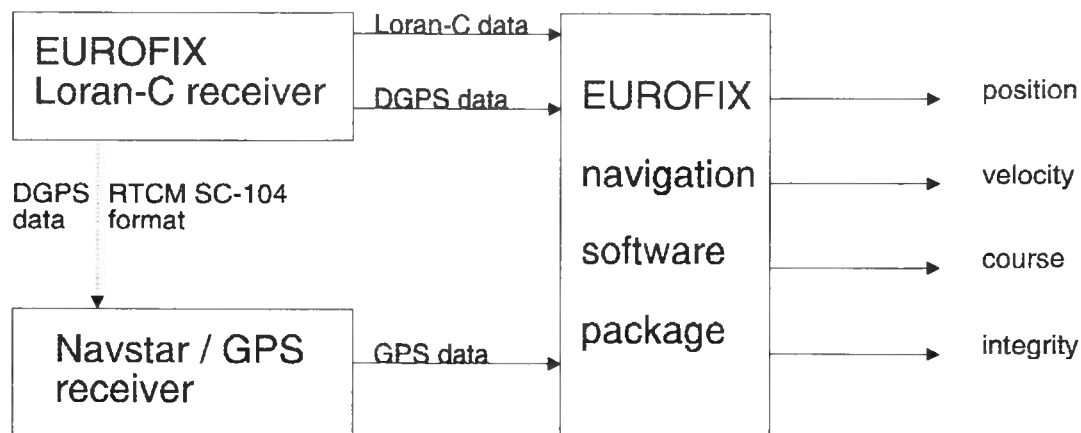


Figure 2.2

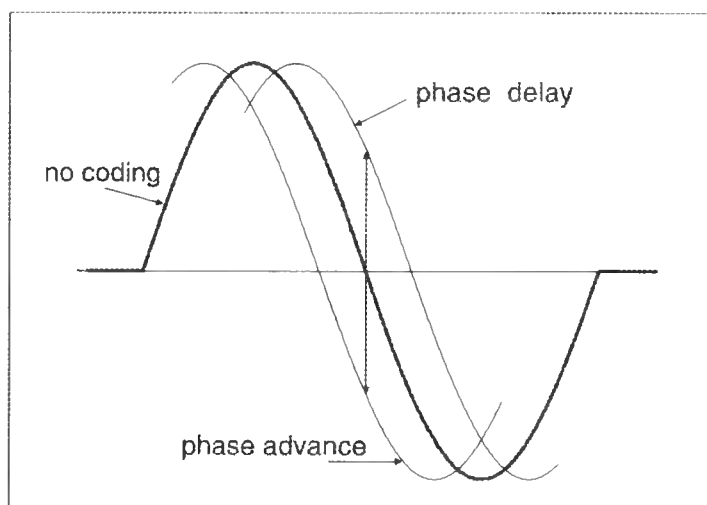
The Eurofix Loran-C receiver generates Loran positions. It also decodes the DGPS data from the Loran-C signal. This DGPS data might be available at an RTCM SC-104 compatible output. The DGPS output could also be connected to some "navigation package", that could take care of integrity monitoring and hybridisation.

This proposed cost-effective Eurofix navigation system offers increased accuracy and integrity over either GPS or Loran-C separately, without the need for a separate data channel that uses additional bandwidth in the scarce radio spectrum.

The objective of Eurofix is to comply to the specified accuracies in the Federal Radio navigation Plan for Harbour Approaches.

2.3 The Eurofix data link

In Eurofix, the data link is constituted by modulation of the Loran-C signal. In the original Eurofix proposals by Van Willigen [12], the last six bursts of the group of eight bursts are shifted with respect to each other, in the following way:



data bit	Modulation pattern							
1	0	0	+	-	+	-	+	-
0	0	0	-	+	-	+	-	+

Modulation pattern for code rate of 6

0 = no time shift

+ = positive code bit = 1 μ s phase delay

- = negative code bit = 1 μ s phase advance

Burst 3..8 of each transmitter are shifted 1 μ s. Either they come 1 μ s in advance, or they will be delayed 1 μ s. In the table above, a 1 μ s early burst is denoted by a -, a burst that is delayed 1 μ s is denoted by a + character. Data bits can be transmitted by applying different modulation patterns to the last 6 bursts. In this way it is possible to transmit 1 bit per station per GRI. This limits the data rate to 10 to 25 bits per second.

Conventional Loran-C receivers will not notice this modulation, as long as the number of burst advances equals the number of burst delays. The modulation effects are averaged out in this way, assuming that the lowest modulation frequency is above the bandwidth of the tracking loop of the receiver.

Transmission of information via the transmitted signals of the Loran-C system is no new concept. The US Coast Guard proposed a two-pulse Loran-C modulation system for transmitting operational information between Loran-C transmitters [9]. And a system called Clarinet-Pilgrim made use of 6 burst for transmitting data.

2.4 Why Eurofix?

The Eurofix system offers users at least 8...20 meter accuracy [10], so that Eurofix can be used for Harbour Approaches. Eurofix is an economic system: a user only needs a slightly modified Loran-C receiver to have DGPS data at his disposal. Eurofix is economic on the transmitter side as well. Only a few transmitters need to be slightly modified to offer a large DGPS service area. Eurofix users might even use (weighted) corrections of a few Eurofix transmitters in one or more chains.

Eurofix is an integrated system, which implies a better system integrity. During normal operations, the Loran-C navigation results will be corrected continuously by DGPS position fixes. If GPS fails, the user still has calibrated Loran-C at his disposal,

so the accuracy will softly degrade to "normal" Loran-C accuracy. The largest Loran-C errors are caused by propagation uncertainties. Once these are calibrated, slowly moving vehicles will be able to navigate with an improved accuracy for some time. If the Loran-C system fails, the user still has standard GPS.

For the reception of DGPS data, no additional data link receiver is necessary. Nor is extra space in the scarce radio spectrum needed to transmit DGPS data, as is the case when radio beacons are used for DGPS data transmission.

These are, in a few words, the big advantages of the Eurofix system. There is one minor drawback: conventional Loran-C users might encounter a slightly degraded Loran-C signal quality. This will be the subject of the next chapter.

3. Eurofix: Painful for conventional Loran-C users?

Information theory states that it is impossible to transmit additional information without increasing the transmission power. Transmitting additional DGPS information using a Loran-C transmitter must cause some deterioration of the navigation capabilities of the Loran-C system when the transmission power remains equal. The consequences of Eurofix modulation for conventional Loran users can (and must be) minimised by several techniques. This will be the subject of this chapter.

3.1 Signal-to-Noise Ratio in Eurofix

As already shown in chapter 2, Loran-C signals are modulated in Eurofix. The modulation consists of shifting burst 3 ... 8 of some station an amount of 1 μ s. The shifting is balanced: three bursts are delayed 1 μ s, three bursts are advanced 1 μ s. For a conventional receiver, those burst shifts should not be visible after averaging.

It is common knowledge that adding 2 sinusoids of the same frequency and amplitude gives another sinusoid of the same frequency. However, the amplitude of the resulting sinusoid depends on the phase difference of the two initial sinusoids:

$$\frac{\cos(\omega \cdot t + \Delta\phi) + \cos(\omega \cdot t - \Delta\phi)}{2} =$$

$$\frac{(\cos(\omega \cdot t) \cdot \cos(\Delta\phi) - \sin(\omega \cdot t) \cdot \sin(\Delta\phi)) + (\cos(\omega \cdot t) \cdot \cos(\Delta\phi) + \sin(\omega \cdot t) \cdot \sin(\Delta\phi))}{2} =$$

$$\frac{1}{2} \cdot \cos(\Delta\phi) \cdot \cos(\omega \cdot t)$$

When a conventional receiver averages modulated Loran-C bursts, it will find a resulting Loran-C burst with a lower signal level, which depends on the modulation

index $\Delta\phi$. Because the noise level will remain the same, a conventional receiver must be content with a reduced Signal-to-Noise Ratio. The problem sketched above can also be made clear by a phasor diagram:

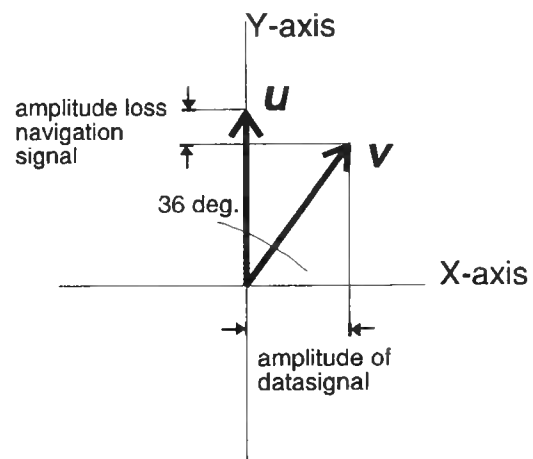


figure 3.1

At some particular zero-crossing, the receiver would normally expect vector u , but as a result of a 1 μ s burst shift, it finds vector v . The navigation component of vector v (the projection of v onto the Y-axis) is smaller than vector u . The quadrature Eurofix data component (projection of v onto the X-axis) of the signal makes data transmission possible. From this figure it is clear that increasing the modulation index towards 90° causes the navigation signal to disappear, but gives excellent data transmission possibilities. The following figure shows how much loss

in Signal-to-Noise Ratio a receiver will suffer from at various modulation indices. This loss not only depends on the modulation index, but also on the number of bursts that is shifted within a group of eight.

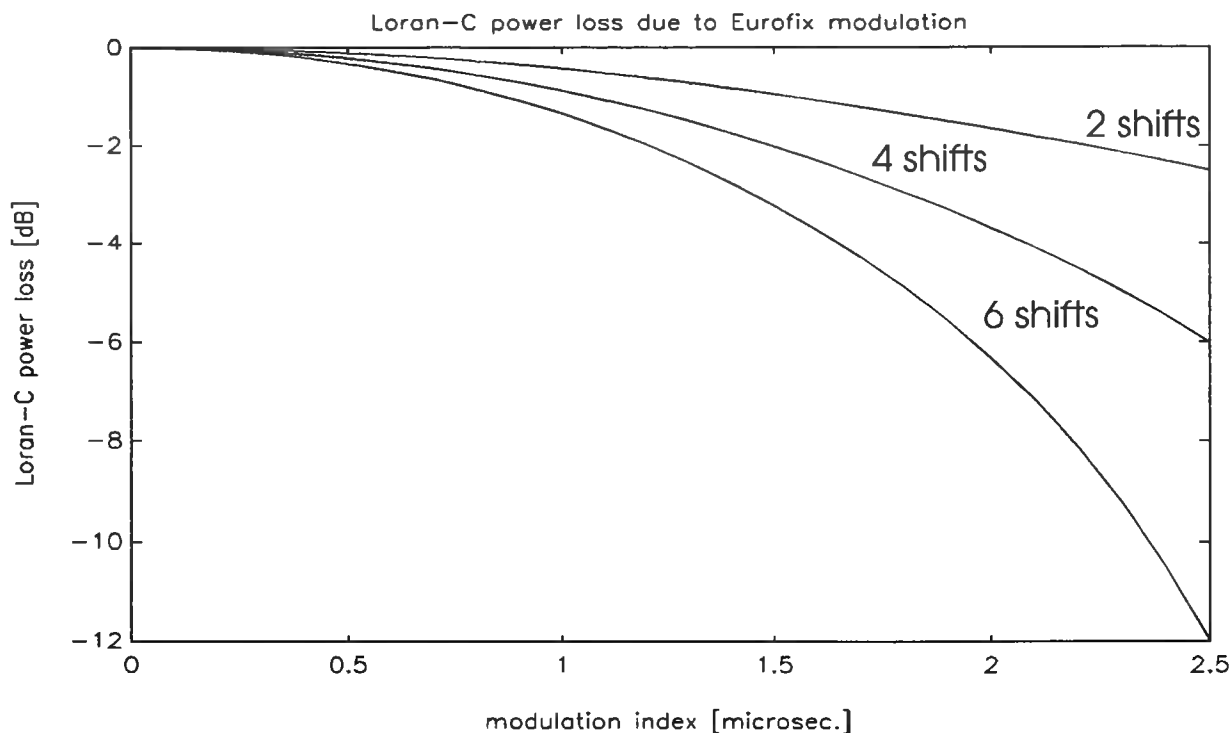


figure 3.2

Assuming that the current generation of conventional receivers cannot track each burst separately and that it will average at least 8 bursts before using these for a position fix, we have the freedom to choose between shifting more respectively. less bursts and a smaller respectively. larger modulation index. However, burst shifting must always be balanced in such a way that the net shift is zero and the burst shifting frequency is larger than the receiver's tracking loop bandwidth.

The above figure shows that the Loran-C navigation power loss is limited. For instance, for the original Eurofix proposals (six burst shifts of 1 μ s each), the loss in signal power is 1.34 dB.

3.2 Zero-crossings of the Eurofix signal

Unfortunately, if the amplitudes of two summed sinusoids are not the same, the zero-crossings of the resulting signal will not be the same, but will be shifted. This also holds for Eurofix signals: because the amplitude of Loran-C bursts is not

constant over the duration of the burst, averaging two shifted bursts will give rise to shifted zero-crossings and in that way to errors in position fixing. When $A_1 \cdot \sin(\omega t + x\Delta\phi)$ and $A_2 \cdot \sin(\omega t - \Delta\phi)$ are added, the resulting sinusoid will have phase:

$$\phi_c = \tan^{-1} \left(\frac{A_1 \cdot \sin(-\Delta\phi) + A_2 \cdot \sin(\Delta\phi)}{A_1 \cdot \cos(-\Delta\phi) + A_2 \cdot \cos(\Delta\phi)} \right)$$

A phasor diagram might clarify this:

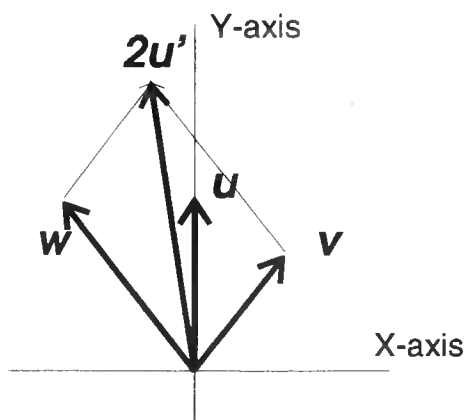


figure 3.3

Above figure demonstrates that the addition of v and w resulting in vector u' doesn't have the same phase as u , although the phase of w is exactly the opposite of the phase of v . Adding two Loran-C bursts is even more complex. A Loran-C burst is described by the following mathematical equation:

$$e(t) = A \cdot \left(\frac{t}{65}\right)^2 \cdot \exp\left(\frac{2-2 \cdot t}{65}\right) \cdot \cos(\omega \cdot t + PC)$$

In this equation, A is not of interest, neither the phase code, assuming that a receiver has already locked onto the signal, so we start with

$$e(t) = \left(\frac{t}{65}\right)^2 \cdot \exp\left(\frac{2-2 \cdot t}{65}\right) \cdot \cos(\omega \cdot t)$$

The result after averaging two bursts that are shifted respectively 1 μ s in advance and 1 μ s later, will be:

$$\frac{e(t + \Delta t) + e(t - \Delta t)}{2}$$

If we discard the factor 2, the averaged Loran-C burst can be described by

$$\left(\frac{t + \Delta t}{65}\right)^2 \cdot \exp\left(2 - \frac{2 \cdot (t + \Delta t)}{65}\right) \cdot \cos(\omega(t + \Delta t)) + \left(\frac{t - \Delta t}{65}\right)^2 \cdot \exp\left(2 - \frac{2 \cdot (t - \Delta t)}{65}\right) \cdot \cos(\omega(t - \Delta t))$$

or

$$\left(\frac{t^2 + 2\Delta t + \Delta t^2}{65^2}\right) \cdot \exp\left(2 - \frac{2t}{65} - \frac{2\Delta t}{65}\right) \cdot \cos(\omega t + \omega \Delta t) + \left(\frac{t^2 - 2\Delta t + \Delta t^2}{65^2}\right) \cdot \exp\left(2 - \frac{2t}{65} + \frac{2\Delta t}{65}\right) \cdot \cos(\omega t - \omega \Delta t)$$

Using the trigonometric relation

$$\cos(\alpha + \beta) = \cos(\alpha) \cdot \cos(\beta) - \sin(\alpha) \cdot \sin(\beta)$$

we can isolate the time shift in the carrier, which yields:

$$\left(\frac{t^2 + 2\Delta t + \Delta t^2}{65^2}\right) \cdot \frac{\exp\left(2 - \frac{2t}{65}\right)}{\exp\left(-\frac{2\Delta t}{65}\right)} \cdot (\cos(\omega t) \cdot \cos(\omega \Delta t) - \sin(\omega t) \cdot \sin(\omega \Delta t)) + \left(\frac{t^2 - 2\Delta t + \Delta t^2}{65^2}\right) \cdot \exp\left(2 - \frac{2t}{65}\right) \cdot \exp\left(\frac{2\Delta t}{65}\right) \cdot (\cos(\omega t) \cdot \cos(\omega \Delta t) + \sin(\omega t) \cdot \sin(\omega \Delta t))$$

The result after averaging contains all kinds of unwanted terms. It is obvious that the zero-crossings of above signal are not the same as would be if no shifting was applied to the bursts. The relations above hold for Loran-C signals *before* filtering. Filtering this expression makes things even more complicated.

But is it really that bad? If we compare the numerically evaluated zero crossings of 8 averaged "normal" Loran-C bursts and eight averaged Eurofix shifted bursts (6 burst shifts, 2 bursts not shifted) after filtering by a 5th order Butterworth filter with $f_c = 100$ kHz and $B = 20$ kHz, we notice that the tracking offset, introduced by Eurofix, is below the resolution of many receivers (100 ns) for zero-crossings above 45 μ s. Appendix A contains a table with zero-crossing offsets for various zero-crossings in appendix A (table A-1).

3.3 Early-late ratios of Eurofix modulated Loran-C signals

The conventional receiver will not only suffer from a degraded Signal-to-Noise Ratio, but, as can be expected, the early-late ratios will also differ when a Eurofix modulation scheme is applied to a normal Loran-C signal. This might cause difficulties for linear Loran-C receivers that use those early-late ratios for cycle selection. However, from table A-2 in appendix A you can make up that difficulties in cycle selection due to Eurofix are not very likely to happen. Again, this table was calculated from 8 averaged Loran-C of which 6 were shifted 1 μ s.

The differences between normal ratios and the ratios that are contaminated by Eurofix modulation are so small that a receiver, especially under bad SNR conditions, will not be able to detect them.

3.4 Minimisation of Eurofix nuisance for conventional Loran-C users

From the previous pages, it will be clear that hindrance for normal Loran-C users cannot be avoided. Even worse: better Eurofix performance implies worse Loran-C performance. Figure 3.1 and figure 3.2 demonstrate that larger burst shifts (a larger modulation index) increase the Eurofix data signal amplitude, and decrease Loran-C power for navigation purposes at the same time.

Loss of signal power can be kept within bounds by controlling the modulation index, or by controlling the number of burst shifts in one GRI. An idea might be to use ternary coding (a +1 μ s or -1 μ s, or no shift at all). When every code symbol (-1, 0 or 1) is equally probable, the average number of bursts that is shifted within one GRI is only 4. Of course, the Eurofix receiver will have to have smaller discrimination margins, so the bit-error-rate will be larger.

Loss of Loran-C signal power is something that cannot be avoided. But tracking errors due to Eurofix modulation can be minimised. There are two possibilities to achieve this. First of all, asymmetrical burst shifting can be implemented. For a certain zero-crossing, the amplitude of the shift forward will then be equal to the shift backward, so the resulting zero-crossing can be found at the place a normal receiver would expect it.

A manufacturer of Loran-C transmitters has already proposed to shift bursts 0.98 μ s forward and 1 μ s backward (delay). This 0.98 μ s value was determined empirically by tests on a hard limiter receiver. A more exhaustive survey will be necessary to determine whether this value also fits other receivers. A disadvantage of this method is that it is only possible to compensate for *one* zero-crossing. Of course, the standard zero-crossing will be chosen for this. But unfortunately, for intelligent receivers that can track zero-crossings that are appropriate under its present Signal-to-Noise Ratios, this is no solution. Another disadvantage is that it is only possible to compensate the offset exactly for 1 particular band-pass filter. The shape of the burst after filtering strongly depends on the filter characteristic. Narrow band-pass filters will smooth the slope of the envelope, so the zero-crossings and ratios effect will be less than with wider band-pass filters, so, how to choose the asymmetrical values?

A better idea is to shift only the carrier with respect to its envelope and leave the envelope at its original position in time, instead of shifting carrier and envelope simultaneously. When the envelope is at its original position, the carriers that will be averaged will have the same amplitude, namely the value of the envelope at that position. The Loran-C receiver then finds an unaffected Loran-C burst (however, at a slightly reduced signal level)! The big advantage of this method is that a normal Loran-C receiver finds all zero-crossings at

the positions where it should find them. The early-late ratios will also be the same. This method compensates all zero-crossing offsets and is independent of filter type.

Eurofix modulation now has become combined ECD Modulation and Pulse Position Modulation instead of pure Pulse Position Modulation. For each burst to be transmitted, the time of transmission will have to be shifted 1 μ s, and an ECD of 1 μ s has to be set in the opposite direction, in order to keep the envelope of the burst at the same time.

Table A-3 in appendix A will summarize tracking offsets for several modulation patterns for a 5th order Butterworth filter with $f_c = 100$ kHz and $B=20$ kHz.

The tracking offsets aren't really that big, because many receivers have a resolution of 100 ns. But these tracking errors come together with tracking errors because of a lower Signal-to-Noise Ratio. If transmitters can be modified to support the method of ECD shifting, it is certainly preferable.

3.5 Conclusion: Eurofix is not painful for conventional Loran-C users

Assuming that offsets in the zero-crossings can be avoided, the conventional Loran-C users only suffer from a slight reduction in Signal-to-Noise Ratio of about 1.4 dB, when the original Eurofix proposals are implemented. Is a loss of 1.4 dB a big problem?

It can be doubted whether a user at the edges of the Loran coverage area will be able to notice this reduction. Besides, at the boundaries of the coverage area, the user will have to switch to some other means of (electronic) navigation anyway. Furthermore, if we take a look at the boundaries of the Loran-C coverage area: is it really necessary to navigate at a 200 meter accuracy in the middle of the Atlantic

Ocean? If so, those users will need some other way of navigation as well.

Loran-C receivers that are modified to support Eurofix, don't suffer from signal degradation. These receivers can notice burst shifts and can take advantage of this knowledge while averaging Loran-C bursts for navigation. Under bad SNR conditions, a Eurofix receiver might take a wrong decision about one particular burst shift and find a range with a large error. However, these wrong decisions can be "filtered" out by using the knowledge on other burst shifts in the same GRI, and by using the knowledge of the track history. Wrongly decoded burst shifts might then be identified and corrected.

If offsets in the zero-crossings cannot be avoided because of technical difficulties in the transmitter stations (when the ECD cannot be adjusted for each separate burst), one could wonder whether that is a real problem. If all Loran-C transmitter stations are equipped with a Eurofix modulator, a conventional receiver will find the same offset in signals coming from all stations. Receivers that use hyperbolic navigation (time-differences!), which will be most Loran-C receivers, will not notice a common offset!

The following chapter will show what a Eurofix user gets in return for the small and hardly noticeable sacrifice of Loran-C signal power. As mentioned earlier, the Eurofix performance depends on the Loran-C Signal-to-Noise Ratio deterioration. A sacrifice of 1.4 dB in power loss might be very well defensible. This 1.4 dB margin enables the Eurofix system to comply to the rules of the Federal Radio navigation Plan for Harbour Approaches. With a margin of 1.4 dB a Eurofix GPS receiver can achieve 8...20 meter accuracies! Better performance is possible at the cost of more Loran-C power. The authorities will eventually decide.....

4. Eurofix: data link valuable for DGPS users?

This chapter will describe the performance of the data link for various modulation schemes. The performance of the data link eventually limits the performance of the DGPS system.

4.1 Introduction

The phrase "performance of the data link" has shown up a few times. An exact definition of "performance" is not trivial, but **data rate** and some kind of **bit-error-probability** or **message error probability** would be part of such a definition. These are the parameters that will be investigated in what is to follow.

Data rate

The data rate of the data link is the number of binary information digits that the data link can transport per second. In the original Eurofix proposal, as sketched in chapter two, 6 bursts of a group of 8 were modulated to transmit *one* bit. So each Loran-C station can transmit 1 bit per GRI. Depending on the GRI (which can last 40 to 100 ms), a Loran-C transmitter can transmit 10 to 25 bits per second. But other modulation patterns might be feasible as well. The use of only 2 burst shifts per bit could also be feasible. In that way, 3 bits can be transmitted per GRI. A burst doesn't have to have 2 possible position shifts, but can have 3 or more as well. Using 2 bursts (remember that the modulation scheme must be balanced!), it is possible to transmit more than 1 bit. The first two bursts of the Loran-C signal may never be modulated. These bursts are used for blinking purposes. Modulating these burst will be an attack on system integrity! A couple of modulation schemes will be discussed later on. To summarize: the eventual data rate of a Eurofix transmitter depends on the GRI (fixed) and the modulation pattern (design parameter).

Bit-error-probability

The counterpart of the data rate is the bit-error-probability. In nearly all applications, one can make a trade-off between a data rate and bit-error-probability. The same is true for Eurofix: we can achieve a larger data rate by allowing bursts to have multiple valid positions. The margins with which we can decide how large the burst shift was, will become smaller and the probability that (due to noise) the wrong decision is made, increases. Burst shifts are decoded by sampling a burst at some specific time. The sample value will tell us the burst shift, which enables us to decode the transmitted information. This sample value will be contaminated by gaussian noise, atmospheric noise (lightning discharges), skywaves, cross-rate interference, carrier wave interference.

Raw bit-error-probability

To give an impression of the Eurofix data transmission properties, the following page contains a figure in which the "raw bit-error-probability" is drawn, as a function of Signal-To-Noise Ratio of the Loran-C signal and the modulation index. As modulation scheme, the original Eurofix proposal (6 burst shifts to transmit 1 bit) was used. By "raw bit-error-probability" the probability that a bit is received erroneously is meant. No knowledge on framing, error detection or error correction is used. The term *bit-error-probability* can also be used later on to define the probability that a bit is wrongly received *after* some error-correction operation. The method of calculating bit-error-probabilities in Eurofix is described in Appendix B.

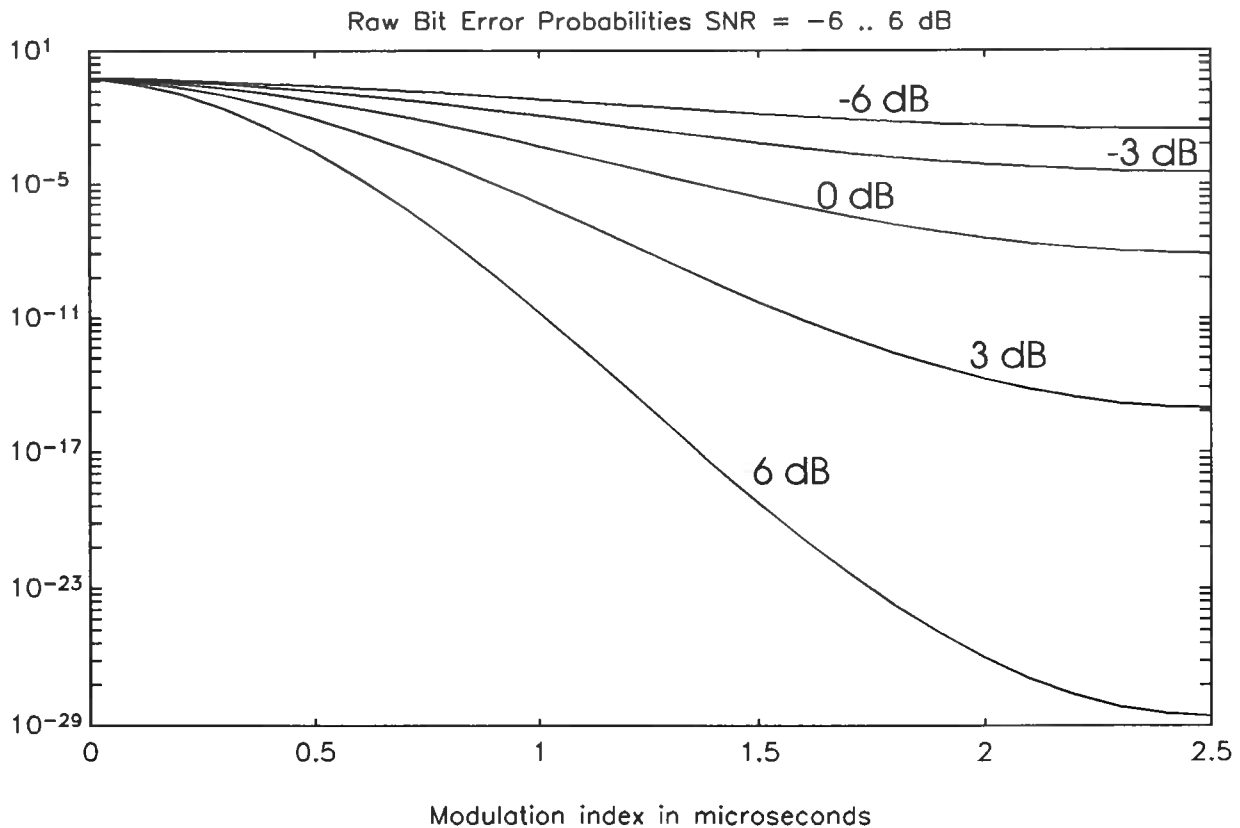


Figure 4.1

From this figure we can draw the premature conclusion that reliable data transmission at -6 dB will be difficult, and that data transmission at 6 dB will cause no problems. Designing a data transmission systems that will operate optimally under both conditions will be a tough job.

4.2 The Eurofix DGPS service area.

Eurofix is primarily meant to supply DGPS data to users, with which they can comply to the rules set in the Federal Radio navigation Plan for Harbour Approaches: professional shipping users so to speak. These rules state that a user should have navigation equipment at his disposal with which he can navigate with an 8 to 20 meter accuracy. These accuracies are needed for *harbour approaches*. Taking a map of the Loran-C service area, we can draw the Eurofix service area roughly by

drawing circles around each Loran-C transmitter. If these circles have a radius of 600 kilometres, all major see-harbour (port of London, Rotterdam, Antwerp, Bremen) and fairways (The Channel, the Thames) will be serviced.

Within this service area, a Signal-to-Noise Ratio of at least 3 dB can be expected, so a SNR of 3 dB might be considered a fair demand for Eurofix. This is the value with which to start the analysis of the Eurofix data link performance. Furthermore, The analysis will also start with a maximum range of 600 km around each transmitter. The data link performance depends on the filter used in the receiver. An analysis for every filter one may think of, would be too time consuming. Instead, the analysis was done using the "standard" filter: a 5th order Butterworth with a centre frequency $f_c = 100$ kHz and a bandwidth $B = 20$ kHz.

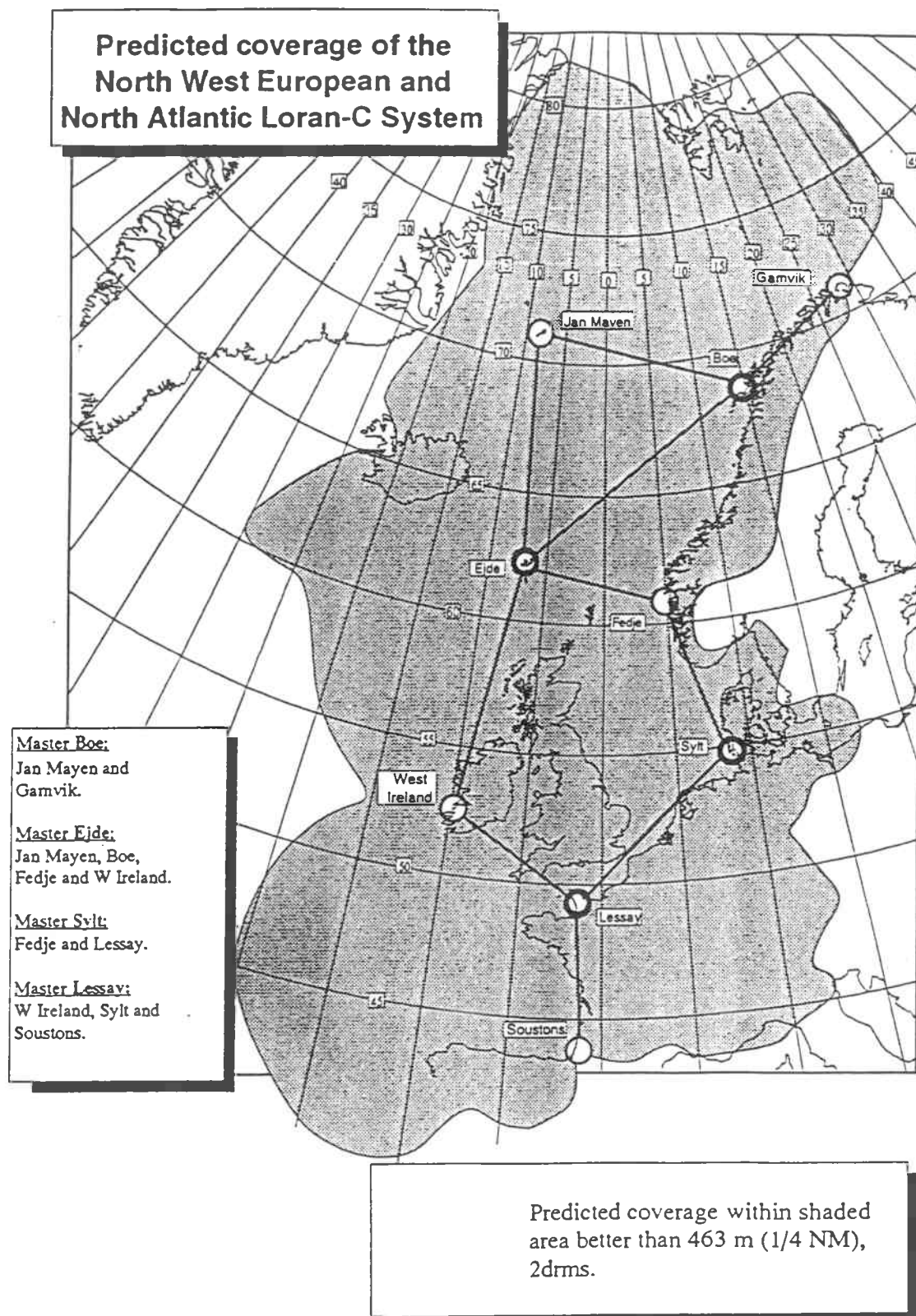


Figure 4.2

For Eurofix we take samples at a specific zero crossing. Of course, we want to take these samples as near to the top as possible: a $1\text{ }\mu\text{s}$ shift at for instance the $125\text{ }\mu\text{s}$ zero-crossing gives better Signal-to-Noise ratios for the data signal (see figure 4.3). Normal navigation using that zero-crossing is not possible, because that zero-crossing will be

severely contaminated by skywaves. But for data transmission, this won't have to be a problem: skywaves are modulated $1\text{ }\mu\text{s}$ as well! If the samples around that specific sample point will be alternating positive and negative under any skywave condition that can be expected, it will be no problem.

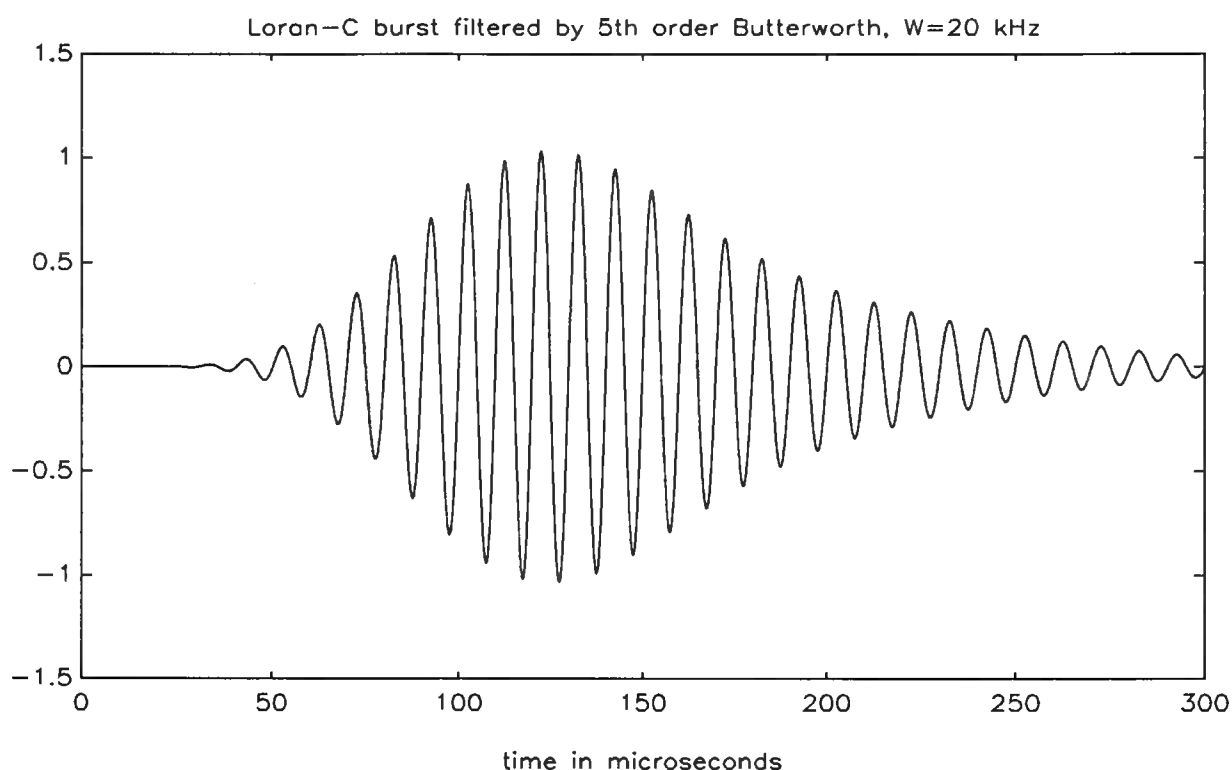


Figure 4.3

Figure 4.4 shows the relationship between groundwave level, skywave level and distance to transmitter. Figure 4.5 shows the relationship between distance and skywave delay. From these figures, we can determine what skywave conditions we can face in the Eurofix service area of 600 km around each transmitter.

Within the Eurofix service area, the earliest skywave we can face is at $60\text{ }\mu\text{s}$ (during daytime). During nights, the earliest skywave we can find is at $85\text{ }\mu\text{s}$. This skywave will be stronger than the skywave during daytime.

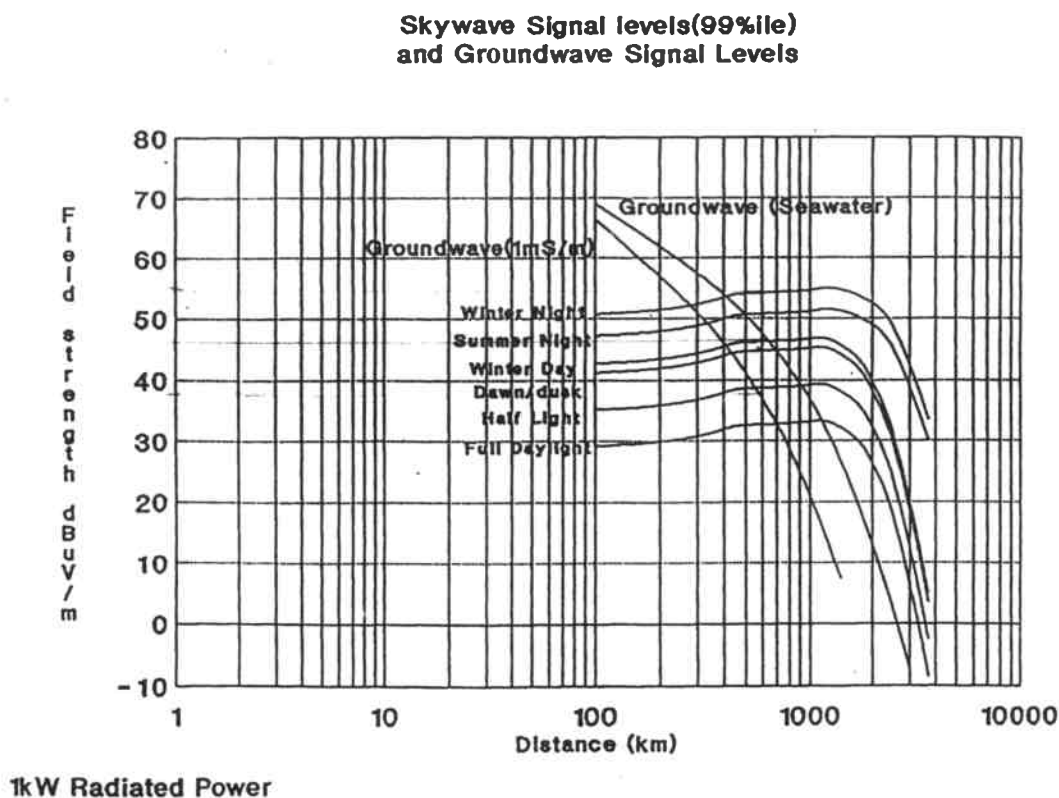


Figure 4.4 (from [8])

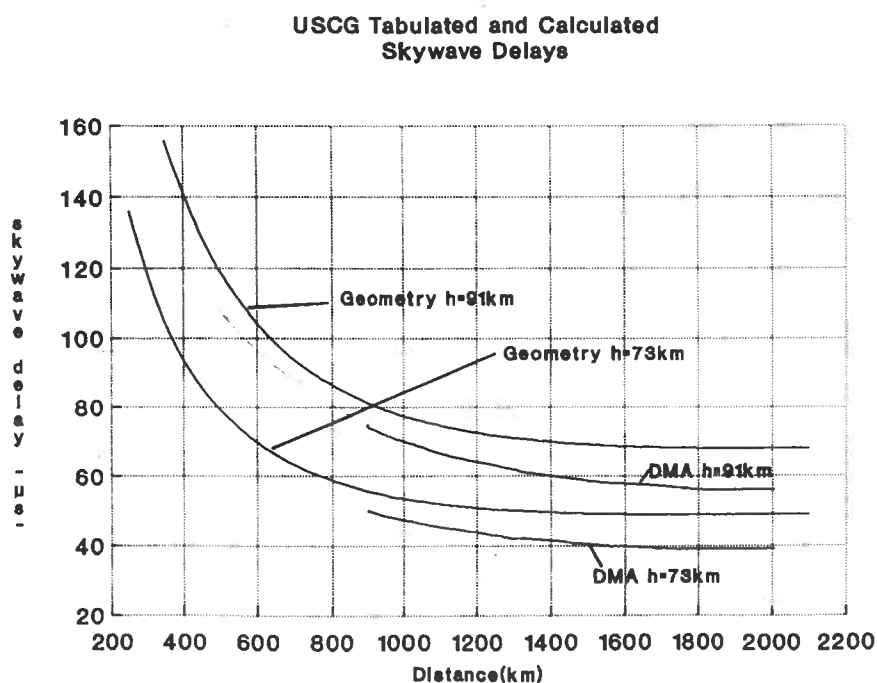


Figure 4.5 (from [8])

Skywaves contaminate the shape of the burst. At a few tens of microseconds after the start of the burst, we can be sure that the burst is not contaminated by a skywave.

The sampled values in this region will be only small, so we will be sensitive for (gaussian) noise then. Sampling later in the bursts gives better Signal-to-Noise Ratios

for the data signal, but also skywave contamination. To determine the influence of skywaves of various levels and with various delays, samples of such contaminated bursts were taken.

The tables in appendix C will show the samples taken at 97.5 μ s and 102.5 μ s for various skywave delays and skywave levels. The samples that are shaded will not be encountered in the Eurofix area. Late skywaves during daytime for instance only have a small level, so these will not be encountered. These tables show that under any valid skywave condition, the early samples (at 97.5 μ s) are all positive, whereas all samples at 102.5 μ s are negative. The absolute magnitude of a sample is at least 0.748. This is the amplitude of the data signal when a burst is modulated with a modulation index of 2.5 μ s. For the calculation of the bit-error-probabilities of figure 4.1, the amplitude of the data signal was

$$0.748 \cdot \sin\left(\frac{\text{index}}{10} \cdot 360^\circ\right).$$

At the 100 μ s zero-crossing, skywave influence is limited. If we take above data signal amplitude as a worst case, we don't have to bother on skywaves with regard to bit-error-probabilities.

Analysis of other (later) zero-crossings has shown that skywave influences are unpredictable. Samples around 100 μ s will give different signs under different skywave conditions. At $t = 100 \mu$ s, the skywave influences aren't that big that data detection is impossible. At $t = 125 \mu$ s they can be. The tables below do not show skywaves later than 85 μ s. Later skywaves don't have influence at $t = 100 \mu$ s. Besides, late skywaves can only be expected at short ranges, and at these ranges groundwave - to-skywave ratios are large. The 100 μ s zero-crossing will be a fair choice for our data link. However, improvements can be achieved when a receiver takes two samples within one burst. The sample with

the largest (absolute) amplitude should then be chosen.

To summarize: For a 5th order Butterworth filter we take the 100 μ s zero-crossing (after filtering). This guarantees a good SNR of the data signal. Sampling after 100 μ s would be preferable, but after that point, the skywave influence will be unpredictable.

4.4 Forward Error Correction

The need for forward error-correcting codes on the data link is somewhat less than for standard RTCM SC 104 DGPS. In standard SC 104 DGPS, one message consists of a *set* of pseudo-range corrections for a few satellites. This message consists of 500 bits of information. The pseudo-range corrections of these satellites must be applied at the *same time*. A bit-error in the pseudo-range correction of *one* satellite ruins the complete messages of 500 bits.

In the Eurofix asynchronous smart DGPS system [10], satellite pseudo range corrections will be transmitted separately. Every satellite pseudo-range correction is independent of other PR-corrections. So, when the user encounters an error in one satellite correction of about 50 bits, only this (small) message has become useless. The Eurofix asynchronous DGPS system is far less sensitive to faulty messages.

However, we can take advantage of Forward Error Correcting (FEC) codes. To illustrate this: if we use ternary coding (see also Appendix A) to transmit 1½ bit per GRI at a modulation index of 1 μ s, and at a SNR of 3 dB, the bit-error-probability is 0.0287 (almost 3%). The probability that a message of 63 bits is transmitted correctly is $(1 - 0.0287)^{63} = 0.16$. One out of 6 messages is received correctly! Using a Reed-Solomon Error Correcting Code that can correct up to 4 errors in a string of 63 bit, the probability that a message is received correctly is increased to 0.9654, at the cost of only a few extra bits.

Atmospheric noise (also known as spherics) is caused by lightning strokes. Several additional strokes, occurring approximately 50 ms apart, may accompany a main stroke and generate a sequence of spikes. Repeated strokes are called multiple discharges and have been observed to consist of 20 to 30 strokes lasting for fractions of a second. These multiple discharges give rise to channel memory, because the channel "remembers" that a stroke has just occurred, and is more likely to provide another stroke. This situation contrasts a random error channel, where the probability of a stroke in a given interval does not depend on whether or not a stroke has occurred recently. Atmospheric noise causes bit errors to occur in bursts. Those bursts can last up to 200 ms, and can be encountered every 1.5 second. These are the two "rules of thumb" that will be used during the rest of the analysis.

For an impression of the capabilities of the Eurofix data link, Reed-Solomon codes will be used for error control coding. These block codes are well known and powerful codes for correction of burst errors. The code that will finally be chosen for the Eurofix data link may be some other code. The code that should be selected, depends on the results of research of Cross Rate Interference and Carrier Wave Interference, which can probably do much more damage than atmospheric noise. Depending on the results of research on error correcting codes for DGPS transmission via radio beacons by Prof. Per Enge and Mrs. Dorothy Poppe, a complete other error correcting code (convolutional code) may be used.

Reed-Solomon codes have the following properties:

$$\begin{array}{ll} \text{length} & n = 2^m - 1 \\ \text{dimension} & k = 2^m - 1 - 2t \\ \text{distance} & d \geq 2t + 1 \end{array}$$

in which m and t are integer numbers with $m \geq 3$ and $1 \leq t \leq 2^{m-1} - 1$. This code needs $n-k = 2t$ "parity" bits. The minimum (Hamming-) distance of a code that is needed, can be derived from the following table:

Detect up to t errors per message	$d_{\min} \geq t + 1$
Correct up to t errors per message	$d_{\min} \geq 2t + 1$
Correct up to t errors per message and detect $l > t$ errors per message	$d_{\min} \geq t + l + 1$

The message length of the code used should be as close as possible to the message length of a Eurofix satellite correction. An error in a message would else cause several satellite corrections to be contaminated. A Reed-Solomon code with a message length of 63 ($m = 7$) suits our purpose.

4.5 Performance analysis of various modulation schemes

The previous paragraphs have introduced the factors that influence the performance of the data link. This paragraph will present the capabilities of a couple of modulation schemes. The schemes are divided into two groups: one group in which 6 bursts are used for data transmission, and one group in which 4 bursts are used for data transmission. Using 2 bursts for data transmission turned out to be absolutely uninteresting for Eurofix purposes, so this possibility won't be discussed.

6 burst shifts per transmitter per GRI

This group was subdivided into 4 cases. The first two cases use all six bursts for 1 code symbol. The last 2 cases use 2 bursts for 1 code symbol, so 3 code symbols per GRI are transmitted. These cases are summarised in the table on the next page.

For a fair comparison, for each of those cases, another Reed-Solomon code was used, in order to correct the error bursts mentioned in the last column. The numbers in the last column were calculated for a GRI of 80 ms. As an example the last column of case 3. The case 3 modulation

scheme can transmit 3 bits per GRI. This gives a data rate of $1 / 80 \text{ ms} * 3 = 37.5$ bits per second. An atmospheric noise burst of 200 ms causes 3 GRI's to be garbled, or 9 bits. In a message of 63 bits (lasting 1.68

seconds) every 1.5 second a burst can occur. So the RS code in case 3 will have to be able to correct 2 noise bursts of 9 bits.

	modulation pattern	possible burst shifts	data rate	error correction requirements
case 1	1 x 6 bursts	binary coding	1 bit per GRI	4 bursts of 3 bits
case 2	1 x 6 bursts	ternary coding	1½ bit per GRI	3 bursts of 5 bits
case 3	3 x 2 bursts	binary coding	3 bits per GRI	2 burst of 9 bits
case 4	3 x 2 bursts	ternary coding	4½ bits per GRI	1 burst of 14 bits

If you compare case 1 with case 3, you see that a higher "raw" bit rate also means more error correction bits. Besides that, a code with a larger bit rate also has a larger bit error probability. So what we're really interested in is the *effective bit rate*: the average number of *good* bits per second after the error detector / corrector. From these bits, wrongly received bits and error control bits are extracted.

The notion of effective bit rate leads to the figure shown below. From this figure one can easily see that the modulation index directly affects the possible bit rate of the Eurofix data link.

At the end of the Eurofix data link, a message is received. The error detection and error correction algorithm then decide what to do with the message. Use it, or discard it. From an integrity point of view, it may not be necessary to maximise the stream of valid messages, but to minimise

the flow of invalid messages. As an example, consider the two following cases: the data link can transport 150 messages per second and the probability that a message is received correctly is 0.8, or, the data link can transport 100 messages per second, and the probability that a message is received correctly is 0.99.

In the first case, 20 percent of the messages will be discarded, resulting in an effective data flow of 120 messages per second. In the second case the effective data flow is 99 messages per second. In the second case, the transmitter can be almost sure that an alert message has reached the receiver. In the first case there is quite a large probability that this alert message is lost. Intuitively, one might want to maximise the flow of valid messages. However, from an integrity point of view, the second case might be preferable. A study on system integrity will be needed to resolve this dilemma.

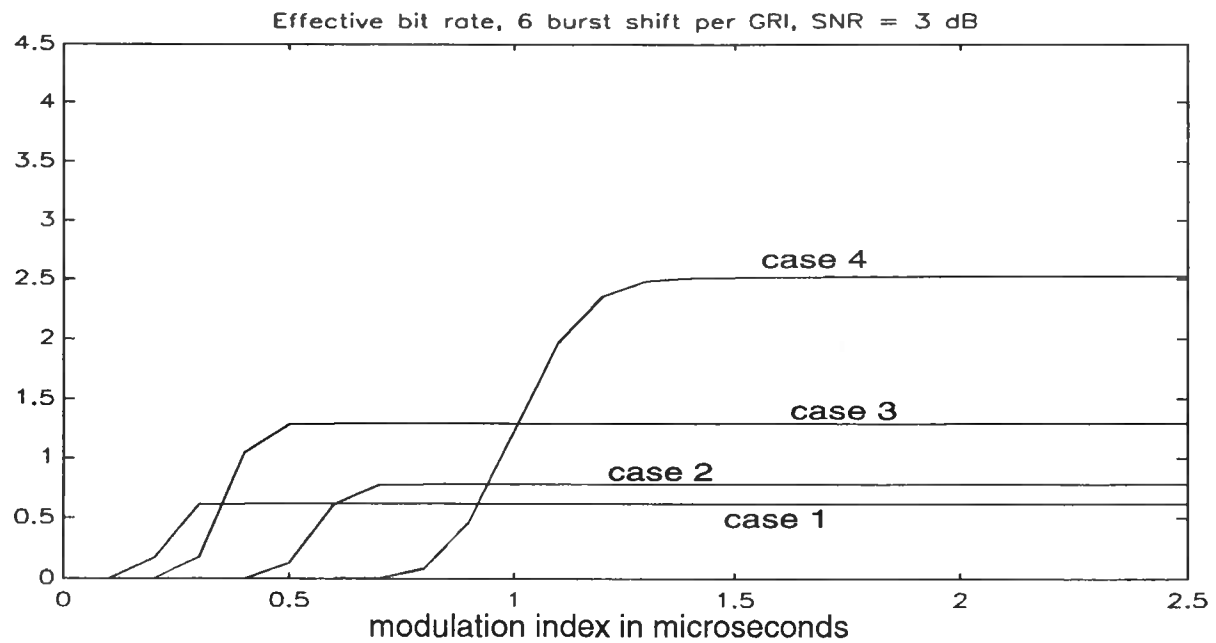


Figure 4.6

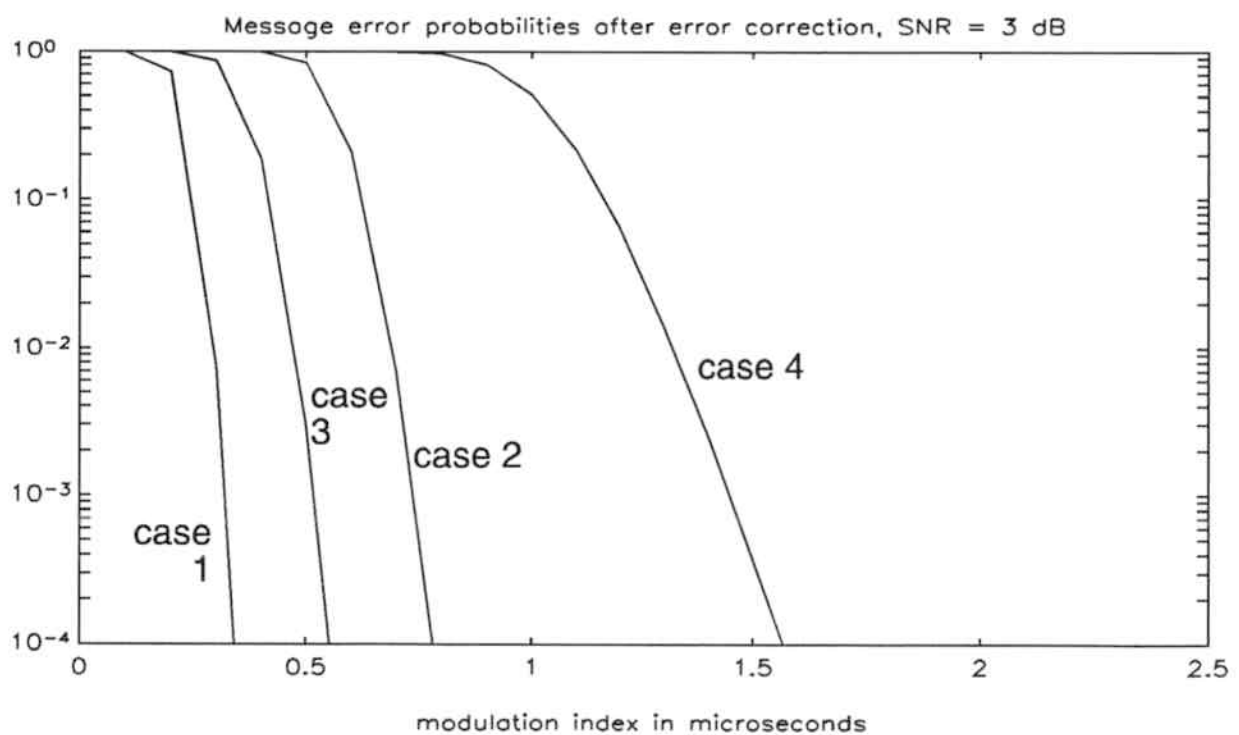


Figure 4.7

For the previous four cases, the probability that a message is received correctly can be read from figure 4.7.

From figures 4.6 and 4.7, the dilemma may become clear. At a modulation index of $1\ \mu\text{s}$, the message error probability is very small compared to the message error probability of case 4 at $1\ \mu\text{s}$. However, the data rate of the modulation scheme in case

4 is quite large compared to the data rate in case 2.

4 burst shifts per transmitter per GRI

This group is subdivided into 4 cases again. The first two cases use four bursts for 1 code symbol. The last 2 cases use 2 bursts for 1 code symbol, so 2 code symbols per GRI are transmitted.

	modulation pattern	possible burst shifts	data rate	error correction requirements
case 5	1 x 4 bursts	binary coding	1 bit per GRI	4 bursts of 3 bits
case 6	1 x 4 bursts	ternary coding	$1\frac{1}{2}$ bit per GRI	3 bursts of 5 bits
case 7	2 x 2 bursts	binary coding	2 bits per GRI	3 bursts of 6 bits
case 8	2 x 2 bursts	ternary coding	3 bits per GRI	2 bursts of 9 bits

The same sort of plots can be made for modulation pattern in which only 4 bursts per transmitter per GRI are modulated. The advantage of modulating only 4 bursts is

that the power loss for the conventional user will be less (see Figure 3.2), or, for an equal conventional user power loss, the modulation index can be larger.

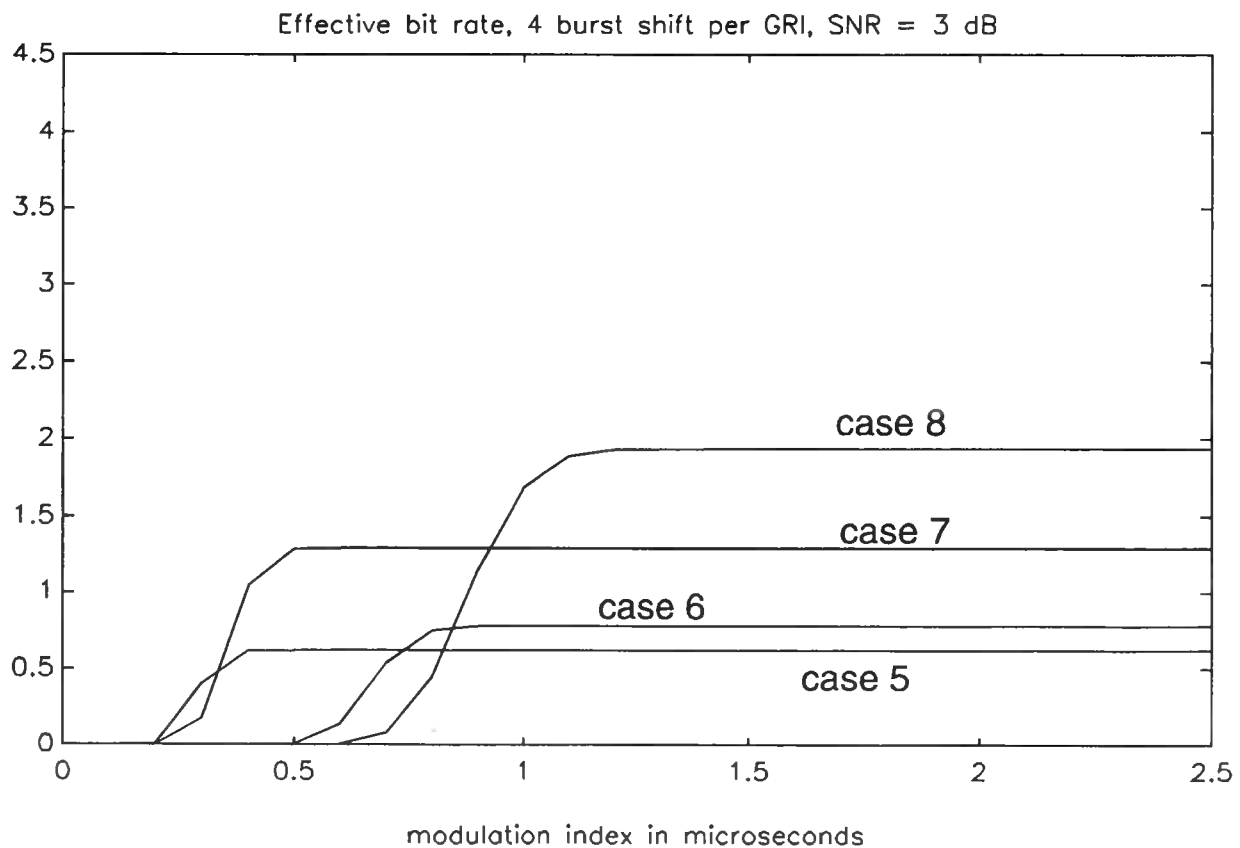


Figure 4.8

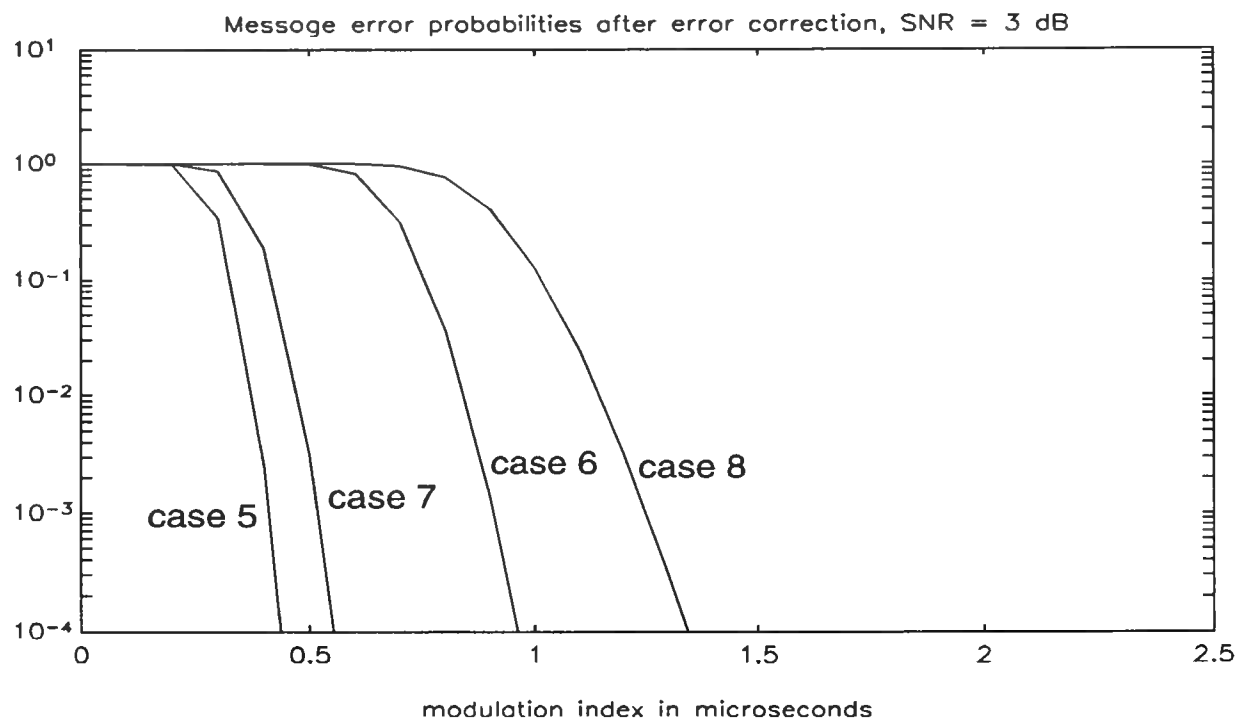


Figure 4.9

Just for comparison purposes you can find the performance plots for 6 burst shifts and 4 burst shift at a Signal-to-Noise Ratio of 0 dB.

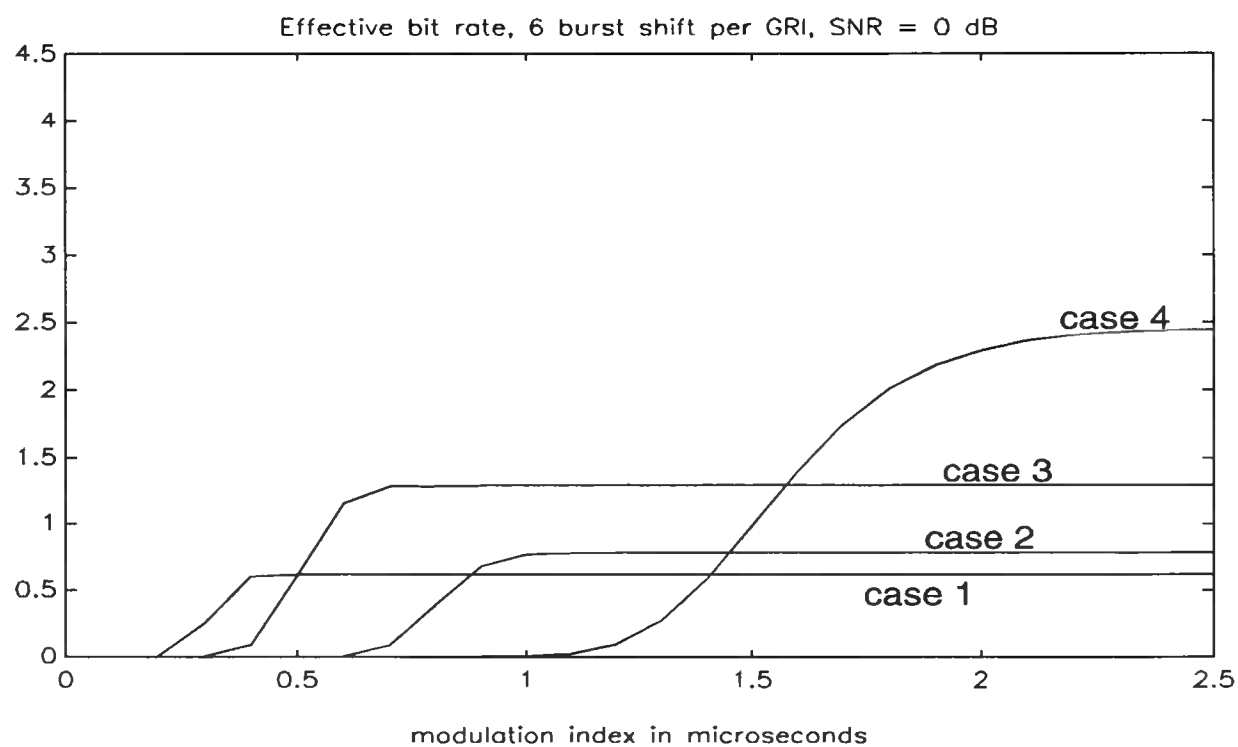


Figure 4.10

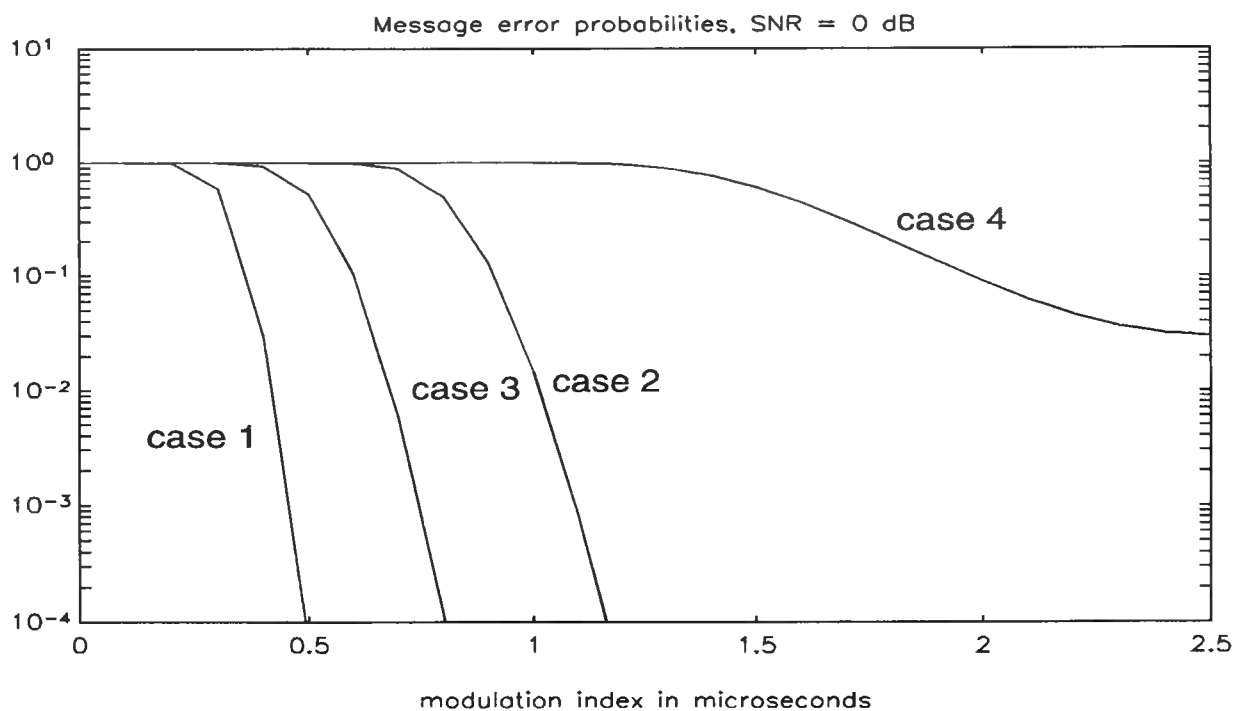


Figure 4.11

Performance plots of 4 modulated bursts at 0 dB:

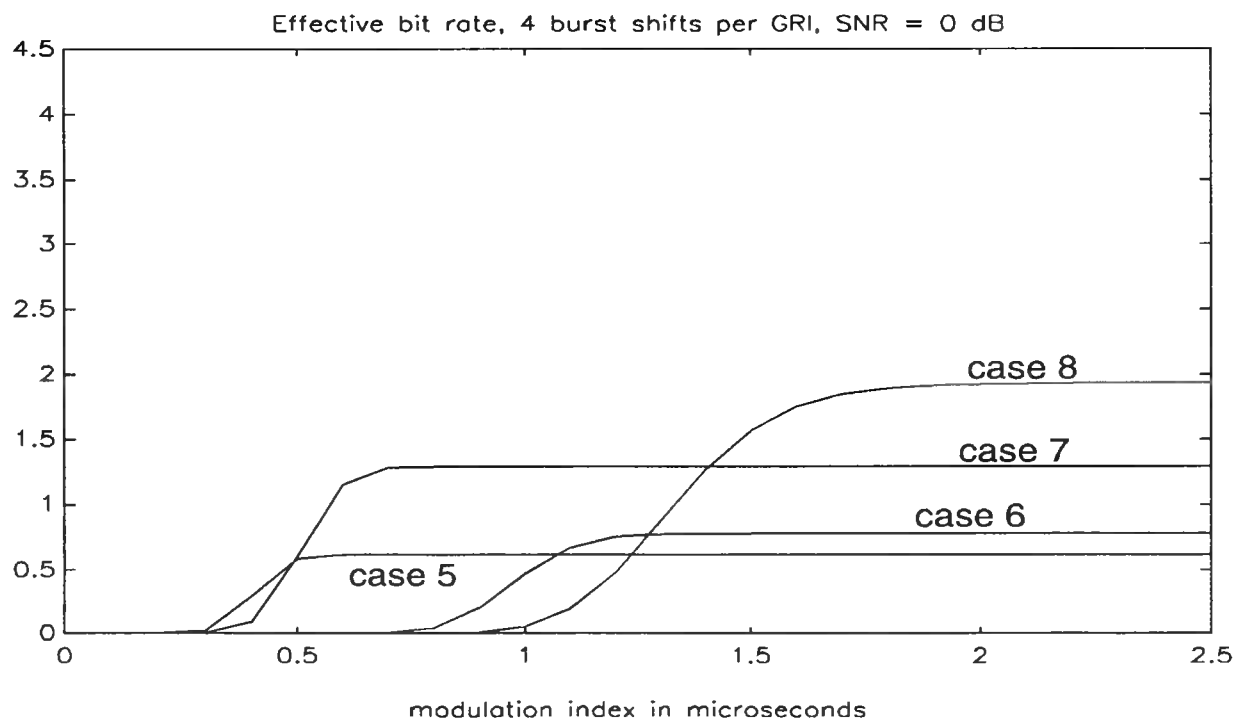


Figure 4.12

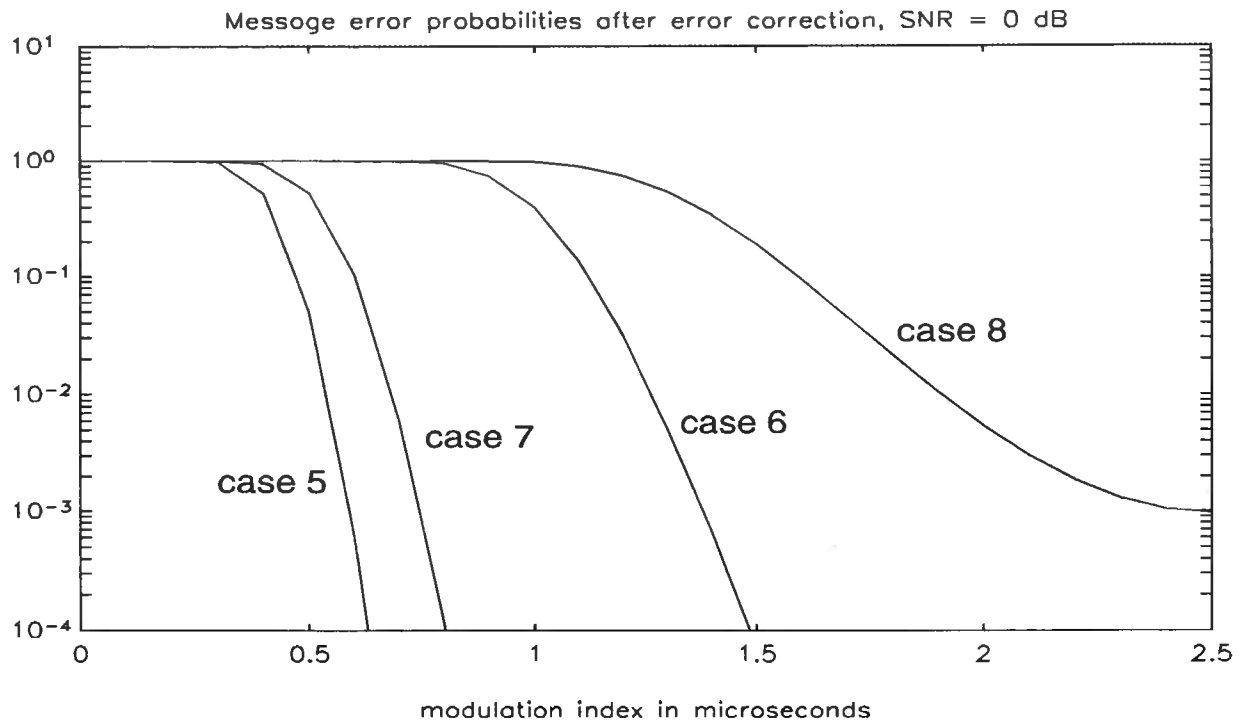


Figure 4.13

4.6 Conclusion: data link is valuable for DGPS users

The previous chapters proved that reliable data transmission by modulating Loran-C is possible. Within the Eurofix service area (prematurely defined as all area's of 600 km around the Loran-C transmitters in Europe), a bit rate of about 1.3 bit per GRI can be obtained by choosing "case 3". In this case, 2 burst within 1 GRI are used to modulate 1 bit. The "raw" bit rate is then 3 bits per GRI. Erroneous messages and error detection and corrections cost 1.7 bit per GRI. The message error probability after error correction is almost neglectible when a modulation index of 1 μ s is chosen. The modulation index can even be smaller to decrease the loss in Loran-C navigation power for conventional users, without affecting the effective bit rate for Eurofix users and without a substantial increase in message error probability.

Above figures were calculated under very pessimistic circumstances of skywaves and atmospheric noise.

The amplitude of the sampled data signal determines the probability that a sample is taken wrongly. In the calculations, a data signal amplitude of 0.748 at a 2.5 μ s modulation was assumed. A skywave with a delay of 64 μ s and a relative strength of 15 dB will cause that the data signal amplitude is that low. In most cases however, the data signal amplitude will be higher and so, the raw bit error rate will be lower.

Above calculations also account for atmospheric noise bursts with a length of 200 ms, occurring every 1.5 second, under the assumption that all bits transmitted during these bursts were received wrongly. Of course, this is also very pessimistic.

So the conclusion that reliable data transmission is possible, is defensible. If the authorities would eventually decide that a modulation index of 1.5 μ s for 6 bursts of a GRI will be acceptable for conventional Loran-C users, even an effective data rate of 2.5 bits per GRI is attainable.

5. Conclusions and recommendations

5.1 Conclusions

Reliable digital data transmission via Loran-C is possible. Depending on what the authorities think is acceptable for Loran-C navigation power loss for conventional Loran-C users, an effective bit rate of 1.3 bit per GRI (16 .. 33 bits per second) can be achieved. The probability that a message contains one or more errors *after* error correction and is discarded for that reason is $1.07 \cdot 10^{-24}$. The probability that a message with one of more error after error correction will pass the error *detector* (that can detect up to 4 errors in a corrected message) is even less.

The modulation principles are flexible. The authorities can make a trade-off between hindrance for "normal" Loran-C users and performance of the data link. This will facilitate the introduction of Eurofix in Europe.

The "pain" for conventional Loran-C users is only limited. They only suffer from a slightly degraded Signal-to-Noise Ratio of 1.34 dB in the case of a modulation scheme in which 6 bursts are modulated, with an amount of 1 μ s. Problems with cycle-identification in linear receivers are not very likely to happen. Offsets in zero-crossings are very small and hardly noticeable for receivers. Besides, if the signals of *all* transmitters of a chain are being modulated for Eurofix, receivers that use hyperbolic navigation (time *differences*) will not be able to notice these offsets as long as each station is tracked on the same zero-crossing and all stations are tracked via the same band pass filter. Eurofix receivers know what they can expect, so they can compensate for the loss in signal power: they just fit an advanced and a delayed burst onto the signal and choose the most appropriate. They can then internally "shift" the burst the opposite direction and use this for an averaged navigation signal.

5.2 Recommendations

Measurement of modulation influence on conventional receivers

Eurofix modulation of Loran-C does not have major implications for the operation of non-Eurofix receivers, as shown in chapter 3. Measurements with conventional receivers and modulated Loran-C signals have to prove this. Measurement results can facilitate the acceptance of Eurofix.

Feasibility study to ECD modulation in Loran-C transmitters

Offsets in zero-crossing cannot be circumvented. But as these offsets are very small and Time Differences won't be influenced in the eventual Eurofix system, these offsets are of only academic importance. However, these offsets can be easily avoided if the transmitter is able to adjust its ECD for every separate burst. It can therefore be recommended to investigate whether it is easy to modify the transmitter in such a way that ECD modulation is possible. If it is not an easy task, the idea of ECD modulation can be abandoned without major implications.

Measurements of the effects of atmospheric noise

Atmospheric noise has been modelled in this report as bit error bursts of a 200 ms length, occurring every 1.5 sec. This is not a realistic model, but a very pessimistic scenario, to demonstrate that Eurofix can work under extreme conditions. At this moment, using this scenario, the system performance will be underestimated. An extensive measurement programme will have to be carried out to adjust the Forward Error Correcting code to the structure of atmospheric error bursts and to give a more realistic system performance.

Research and measurements of the effects of CRI and CWI

This report has not gone into the subject of Cross Rate Interference, nor Carrier Wave Interference. But it is not difficult to imagine that both types of interference will affect the performance of the data link. Research will have to be done to model the influence of Cross Rate Interference and Carrier Wave Interference on data transmission. Together with the result of measurements of atmospheric noise, an appropriate forward error correcting code can be chosen or designed to maximise the effective bit rate and to minimise the remaining message error probability.

Research on the influences of the Eurofix data link on system integrity

In chapter 4 a dilemma was sketched: What is preferable: an effective bit rate of 120, but a message error probability of 80%, or an effective bit rate of 99 and a message error probability of 1%? The choices that will have to be made for the ultimate data link will depend on the answer to above question. A survey into the system integrity of Eurofix might yield this answer.

Calculations on the coverage area

Which areas in Europe are interesting for Eurofix? The assumption that the areas within a range of 600 km around each transmitter cover all interesting parts will have to be verified. Within an area of 600 km around a transmitter, a Signal-to-Noise Ratio of 3 dB was assumed to be present. Probably, the SNR will be better. In that case, do we want to extend the Eurofix service area, or will we use a higher SNR for better performance of the data link? Calculations of SNR's that can be encountered in Europe will have to provide answers on this. The Eurofix coverage area is not the same as the Loran-C coverage area, for only *one* transmitter is needed to receive DGPS data. Potential Eurofix buyers will want to have a map with Eurofix coverage, the authorities will like a map with the conventional Loran-C

coverage area *after* Eurofix modulation. These maps will have to be drawn.

Literature references

- [1] Radio Technical Commission for Marine Services,
Minimum Performance Standards (MPS) Marine Loran-C Receiving Equipment
Radio Technical Commission for Marine Services, Washington D.C. 20554, United States of America, December 20, 1977
- [2] A. Bruce Carlson,
Communication Systems
Singapore: McGraw-Hill, Inc, ISBN 0-07-Y100560-9, third edition, 1986
- [3] Martin Beckman,
Carrier Wave Signals Interfering with Loran-C
The Netherlands: Delft University of Technology, Faculty of Electrical Engineering, PhD Thesis, 1992 .
- [4] Shu Lin, Daniel J. Costello,
Error Control Coding: Fundamentals and Applications
Prentice-Hall, Inc. Englewood Cliffs, New Jersey, USA, 1983
ISBN: 0-13-283796-X
- [5] Arnold M. Michelson, Allen H. Levesque,
Error-control Techniques for Digital Communication
John Wiley & Sons, New York, USA, 1985
ISBN: 0-471-88074-4
- [6] Per K. Enge, Rudolph M. Kalafus, Michael F. Ruane,
Differential Operation of the Global Positioning System,
IEEE Communications Magazine, july 1988, Vol. 26, Nr. 7
- [7] Dorothy C. Poppe,
Burst Error Correction for a Medium Frequency Broadcast of Differential GPS Data,
Institute of Navigation, ION GPS-91, ION Satellite Division's 4 th international Technical Meeting, September/October 1991
- [8] David Last, Richard Farnworth, Mark Searle,
Ionospheric Propagation and Loran-C range - The Sky's the limit
- [9] D.A. Feldman, M.A. Letts, R.J. Wenzel,
The Coast Guard two-pulse Loran-C communications systems,
Navigation, Vol. 23, Nr. 4, pp. 279-288, Winter 1976-1977
- [10] L.J. Beekhuis,
The Asynchronous Correction Concept for High Precision DGPS in Eurofix,
The Netherlands, 1993, Delft University of Technology, Faculty of Electrical Engineering, Telecommunication and Traffic Control Systems

- [11] A.M. Noorbergen,
Simulating the Eurofix Data Link,
The Netherlands, 1993, Delft University of Technology, Faculty of Electrical Engineering, Telecommunication and Traffic Control Systems

- [12] Prof. Dr. Ir. D. van Willigen,
Eurofix, a Synergism of Navstar/GPS and Loran-C
The Netherlands, Delft University of Technology, Faculty of Electrical Engineering, Telecommunication and Traffic Control Systems

- [13] Prof. Dr. Ir. D. van Willigen,
Integrated Eurofix and IALA's DGPS: Improved Integrity and Availability
The Netherlands, Delft University of Technology, Faculty of Electrical Engineering, Telecommunication and Traffic Control Systems

A. Tracking offsets and early-late ratios in Eurofix

Eurofix modulation of Loran-C signals has consequences for "normal" Loran-C navigation. On the one hand, the zero crossings of the averaged Loran-C signals will be shifted. On the other hand, the early-late ratios of the envelope (used for cycle selection in linear receivers) will have different values. These effects are listed in the tables below. The zero crossings and early-late ratios of the Loran-C signal were evaluated after begin filtered by a 5th order Butterworth filter with a centre frequency $f_c = 100$ kHz and a bandwidth $B = 20$ kHz.

Table A-1 lists the zero crossings of 8 "normal" averaged Loran-C bursts, the zero crossings of a Loran-C signal consisting of 8 bursts of which 6 bursts were advanced and delayed 1 μ s alternately and the difference of these zero crossings.

Zero crossings of 8 "normal" filtered bursts in μ s	Zero crossings of filtered Eurofix bursts in μ s	Tracking error in ns
7,6741	7,1760	498,0
12,1398	11,7885	351,4
16,6866	16,4199	266,7
21,3344	21,1231	211,3
26,0632	25,8910	172,2
30,8524	30,7093	143,1
35,6861	35,5655	120,6
40,5529	40,4502	102,7
45,4446	45,3564	88,2
50,3553	50,2793	76,1
55,2810	55,2152	65,9
60,2186	60,1615	57,1
65,1658	65,1164	49,5
70,1211	70,0783	42,8
75,0830	75,0462	36,8
80,0507	80,0192	31,5
85,0235	84,9968	26,7
90,0008	89,9785	22,3
94,9822	94,9639	18,3
99,9673	99,9527	14,6
104,9558	104,9446	11,2
109,9476	109,9396	8,0
114,9426	114,9375	5,1
119,9404	119,9381	2,3
124,9411	124,9413	-0,2
129,9445	129,9470	-2,5
134,9503	134,9550	-4,7
139,9583	139,9650	-6,7
144,9683	144,9767	-8,4
149,9798	149,9898	-9,9

Table A-1

Table A-2 gives the early-late ratios of the envelope around the zero crossings listed in the second column. The third column gives the early-late ratios that a linear type of receiver would encounter when no Eurofix modulation is applied to the Loran-C signal. The fourth column gives the early-late ratios of envelope samples when Eurofix modulation (6 bursts out of 8 are delayed and advanced 1 μ s alternately) is applied.

	"normal" zero-crossing in μ s	early-late ratio at normal zero-crossing	early-late ratio at Eurofix zero-crossing
1	7,6741	25,3639	24,2313
2	12,1398	9,0498	9,0271
3	16,6866	5,2139	5,2375
4	21,3344	3,6734	3,6906
5	26,0632	2,8780	2,8887
6	30,8524	2,4034	2,4101
7	35,6861	2,0924	2,0968
8	40,5529	1,8747	1,8777
9	45,4446	1,7145	1,7167
10	50,3553	1,5921	1,5937
11	55,2810	1,4957	1,4968
12	60,2186	1,4176	1,4185
13	65,1658	1,3532	1,3539
14	70,1211	1,2989	1,2995
15	75,0830	1,2526	1,2531
16	80,0507	1,2125	1,2129
17	85,0235	1,1773	1,1777
18	90,0008	1,1462	1,1465
19	94,9822	1,1184	1,1187
20	99,9673	1,0935	1,0937
21	104,9558	1,0709	1,0711
22	109,9476	1,0504	1,0506
23	114,9426	1,0317	1,0319
24	119,9404	1,0146	1,0148
25	124,9411	0,9990	0,9992
26	129,9445	0,9848	0,9850
27	134,9503	0,9720	0,9721
28	139,9583	0,9604	0,9605
29	144,9683	0,9502	0,9503
30	149,9798	0,9412	0,9413

Table A-2

Table A-3 summarises tracking errors due to Eurofix modulation at the zero crossings listed in the first column. The second column shows the tracking errors that will be encountered when 6 out of 8 bursts are advanced and delayed symmetrically 1 μ s. The third column gives the tracking errors that result from asymmetrical burst shifting (again, 6 out of 8 bursts). Perhaps superfluous, the last column lists the tracking errors that could be encountered when ECD modulation is used in Eurofix as modulation mechanism.

"Normal" zero-crossings in μ s	tracking error in ns 1 μ s adv, 1 μ s delay	tracking error in ns 0.98 μ s adv, 1 μ s delay	tracking error in ns ECD modulation
7,6741	498,0	479,7	0
12,1398	351,4	336,2	0
16,6866	266,7	253,5	0
21,3344	211,3	199,5	0
26,0632	172,2	161,2	0
30,8524	143,1	132,8	0
35,6861	120,6	110,8	0
40,5529	102,7	93,4	0
45,4446	88,2	79,1	0
50,3553	76,1	67,3	0
55,2810	65,9	57,3	0
60,2186	57,1	48,8	0
65,1658	49,5	41,3	0
70,1211	42,8	34,8	0
75,0830	36,8	29,0	0
80,0507	31,5	23,7	0
85,0235	26,7	19,0	0
90,0008	22,3	14,7	0
94,9822	18,3	10,8	0
99,9673	14,6	7,2	0
104,9558	11,2	3,8	0
109,9476	8,0	0,7	0
114,9426	5,1	-2,1	0
119,9404	2,3	-4,8	0
124,9411	-0,2	-7,3	0
129,9445	-2,5	-9,6	0
134,9503	-4,7	-11,7	0
139,9583	-6,7	-13,6	0
144,9683	-8,4	-15,3	0
149,9798	-9,9	-16,8	0

Table A-3

B. Calculation of Bit-Error-Probabilities in Eurofix

B.1. Binary error probabilities

Detecting databits from a Eurofix signal requires sampling at some zero-crossing. Depending on the direction of the burst shift, the sampling circuitry will find a positive or a negative value.

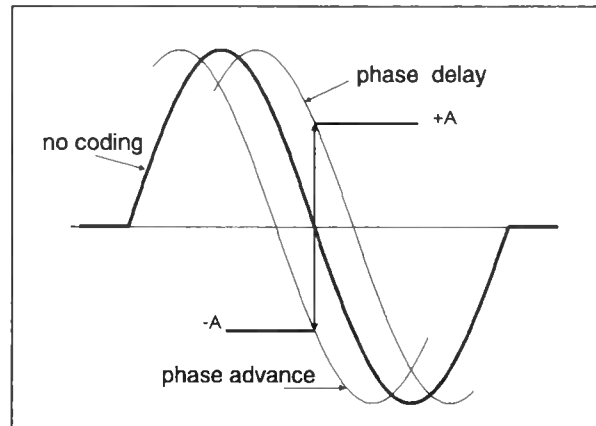


Figure B.1

The value the sampling circuitry should find when the burst is delayed, is $+A$. The sampler should find a $-A$ when the burst is advanced. However, the sampling circuitry only makes distinction between positive and negative values.

The values $-A$ and $+A$ are contaminated with noise. If we consider the noise to be additive white Gaussian noise with zero mean and variance σ^2 , the probability density functions of the samples look as follows:

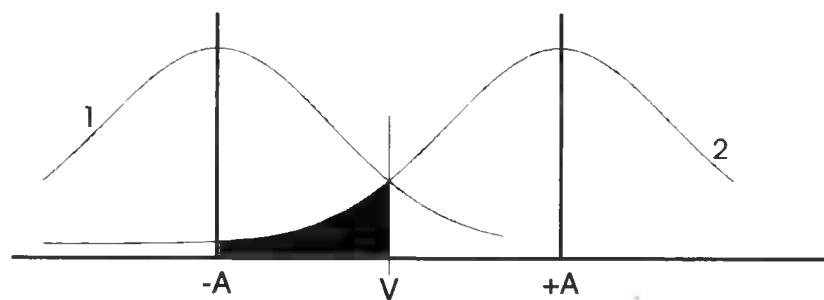


Figure B.2

Curve 1 gives the probability density function of the sample value when an advanced burst is transmitted. Curve 2 gives the probability density function of the value that the sampling circuitry will find when a delayed burst is transmitted. The decision threshold V is taken zero in the Eurofix case: any positive sample will be regarded as a burst delay, a negative value will be regarded as a burst advance. This decision will go wrong, when a burst is delayed (so the sample should be $+A$), but due to a negative noise excursion, the signal is negative. The sampler will in that case wrongly decide that the burst was advanced.

If the probability density function of the noise is $p(n) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-n^2/2\sigma^2}$ the probability that a burst delay is regarded as a *burst advance* is:

$$P_r(\text{burst advance} \mid \text{burst delay}) = P_r(\text{sample} < V) = \int_{-\infty}^{V-A} \frac{1}{\sqrt{2 \cdot \pi \cdot \sigma^2}} \cdot e^{\frac{-x^2}{2\sigma^2}} dx$$

A similar formula can be derived for the probability that a burst advance is regarded as a *burst delay*:

$$P_r(\text{burst delay} \mid \text{burst advance}) = P_r(\text{sample} > V) = \int_A^{\infty} \frac{1}{\sqrt{2 \cdot \pi \cdot \sigma^2}} \cdot e^{\frac{-x^2}{2\sigma^2}} dx$$

Because of the symmetry of the Gaussian pdf, the probability that a burst shift will be decoded erroneously is

$$P_e = Q\left(\frac{A}{\sigma}\right) \text{ in which } Q(k) = \frac{1}{\sqrt{2\pi}} \cdot \int_k^{\infty} e^{\frac{-\lambda^2}{2}} d\lambda$$

Armed with above formula's, we can try to calculate probabilities that a burst shift is decoded erroneously. The A and the σ from above formula's can be derived from the Signal-to-Noise Ratio of a Loran-C signal.

For the definition of signal levels in Loran-C, a quotation of the Minimum Performance Standards of the RTCM [1] will follow:

The level of a Loran-C signal (a group of pulses from a single transmitting station) is the rms level of a CW signal having the same peak-to-peak amplitude as the Loran-C pulse envelope 25 μ s after the beginning of the pulse. The 25 μ s point is referred to as the Standard Sampling Point (SSP). This standard sampling point is used for defining signal amplitude (.....). The level, when discussing electromagnetic fields, is expressed in decibels above one microvolt per meter (dB / 1 microvolt / 1 meter). Otherwise, the level may be expressed in rms volts (V, mV, μ V) at the standard sampling point.

For a "standard" Loran-C envelope with the following parameters:

$$S(t) = A t^2 e^{-2t/65} \quad (\text{for } t \leq t \leq 65 \mu\text{s})$$

the value at the 25 μ s is .506 of peak. Therefore, to convert the rms signal to zero-to-peak voltage for measurement with an oscilloscope, the following formula can be used:

$$V_{op} = \left(\sqrt{2} \cdot \frac{1}{.506} \right) \cdot S_{rms} \quad (\text{at } 25 \mu\text{s})$$

The above definition for Loran-C signal level will apply to cross-rate Loran-C signals as well as desired Loran-C signals

Differential signal level is the difference between the levels of two signal, expressed as the voltage ratio of the larger to the smaller in decibels (dB)

The Minimum Performance Standards state the following on noise:

Noise

Simulated random noise (Gaussian) will be considered to have a uniform power spectral density prior to filtering. After filtering by a single resonator L-C filter having a center frequency of 100 KHz and a 3 dB bandwidth of 30 KHz, the noise level is the voltage generated across a 50 Ω resistive load, measured on a true rms voltmeter; this noise level is defined as the rms noise level, denoted by N.

Signal-to-Noise Ratio (SNR)

The signal-to-noise ratio is the ratio of the Loran-C signal level at the SSP to the rms level of simulated noise. SNR can be expressed as a dimensionless number indicating rms voltage / rms voltage or as a ratio in dB.

By the way, the LC-filter mentioned in the MPS noise definition has a noise equivalent

$$\text{bandwidth } B_n = \int_0^{\infty} |H(j\omega)|^2 d\omega = 47.124 \text{ KHz [3].}$$

With these definitions, we can calculate the noise and signal level in the receiver. The power of the noise in accordance with the MPS at the antenna input is given by the following formula:

$$\sigma^2 = \frac{Env^2}{2} \cdot 10^{\left(\frac{SNR}{10}\right)}$$

By dividing this power by the noise bandwidth of the 30 kHz filter, the noise power density No of the antenna noise can be found:

$$N_o = \frac{\sigma^2}{B_{n, LC - filter}}$$

in which $B_{n, LC-filter}$ the equivalent noise bandwidth of the LC-filter specified in the MPS definitions. This noise bandwidth is 47.124 kHz.

Once the noise power density of the antenna noise is found, the power of the noise after the filter system can be calculated:

$$\sigma_{filtered}^2 = B_{n, filter} \cdot N_o$$

in which $B_{n, filter}$ is the equivalent noise bandwidth of the filter system (band-pass- and notch filters). The noise equivalent bandwidth of our "standard" 5th order Butterworth filter is 20,333 kHz.

B.2. Example of Binary-Error-Probability.

A Eurofix data bit consists of several burst shifts. In the original Eurofix proposal, a bit is transmitted by 6 alternating burst shifts. To calculate the probability of a bit-error, we first need to calculate the probability that a single burst shift is detected erroneously. In this example we take the following parameters:

SNR = 3 dB (MPS definition)

Loran-C bandpass filter: 5th order Butterworth, $f_c = 100 \text{ kHz}$, $B = 20 \text{ kHz}$, $B_n = 20.333 \text{ kHz}$

ZC = 100 μs

Modulation index = 1 μs .

The Loran-C signal will be tracked at the 100 μs zero-crossing. Sampling the signal (without noise) at 100 μs +/- 2.5 μs will give at least a value of 0.748 under any valid skywave condition (see Chapter 4). Sampling that signal at 1 μs from the zero crossing will give

$$A = 0.748 \cdot \sin\left(\frac{1}{10} \cdot 360^\circ\right) = 0.4397.$$

So, assuming a balanced shifting, the amplitude of the *data* signal will be either -0.4397 or 0.4397. For reference: the value of the envelope at 25 μ s (SSP) is 0.506. The top of the envelope is 1.

The noise power of the signal is $\sigma^2 = \frac{Env^2_{25}}{2} \cdot 10^{-\left(\frac{SNR}{10}\right)} = 0.06416099$. The filtered noise power

is $\frac{20,333}{47,124} \cdot 0.06416099 = 0.0276841$. By taking the square root of this, we find σ to be

0.16638538. The probability that a particular burst is decoded the wrong way is now

$$P_e = Q\left(\frac{A}{\sigma}\right) = Q\left(\frac{0.4397}{0.16638538}\right) = 0.00410875$$

If we assume the originally proposed Eurofix coding scheme (6 burst shifts per bit), and decide that a bit is decoded wrongly when 3 or more burst shifts are decoded wrongly (majority vote principle), the probability that a *bit* is decoded erroneously is

$$P_r(\text{bit-error}) = \sum_{i=3}^6 \binom{6}{i} \cdot p^i \cdot (1-p)^{6-i}$$

When p is the *sample-error* probability. Taking $p = 0.00410875$, the *bit-error-probability* will be $P_r(\text{bit-error}) = 1.38 \cdot 10^{-6}$.

When one single Eurofix message consists of 50 bit, the probability that a message contains one or more errors is

$$P_r(\text{message-error}) = \sum_{i=1}^{50} \binom{50}{i} \cdot p^i \cdot (1-p)^{50-i} \text{ in which } p \text{ is the bit-error probability. This}$$

message error probability $P_r(\text{message-error}) = 6.8998 \cdot 10^{-5}$

B.3. Ternary-error-probability

In the original Eurofix proposal, burst 3...8 of a group of eight are shifted either 1 μ s in advance, or will be delayed 1 μ s. Extra information can be transmitted by also allowing a shift of 0 μ s (burst will *not* shift). In this way we can transmit a -1, a 0 or a 1 source digit by shifting the bursts according to the patterns + - + - + -, or 0 0 0 0 0 0, or - + - + - +. This gives us a data rate of 1.5 bit per GRI, at the cost of a larger tri-bit (trit ?) error probability.

The values the sampler can expect are now: -A, 0 and +A. The decision threshold will be set at $-\frac{1}{2}A$ and $\frac{1}{2}A$. In a similar way we derived a formula for the binary case, we can derive:

$$Pr(\text{burst advance wrongly received}) = Q\left(\frac{\frac{1}{2} \cdot A}{\sigma}\right)$$

$$Pr(\text{static burst wrongly received}) = 2 \cdot Q\left(\frac{\frac{1}{2} \cdot A}{\sigma}\right)$$

$$Pr(\text{burst delay wrongly received}) = Q\left(\frac{\frac{1}{2} \cdot A}{\sigma}\right)$$

The total probability that some sample is wrong, is the sum of the probabilities that some "trit" was transmitted, multiplied by the probability that that "trit" was received erroneously, so

$$P_e = \frac{1}{3} \cdot P_e(\text{burst delay}) + \frac{1}{3} \cdot P_e(\text{static burst}) + \frac{1}{3} \cdot P_e(\text{burst advance}) = \frac{4}{3} \cdot Q\left(\frac{\frac{1}{2} \cdot A}{\sigma}\right)$$

with $Q(k)$ still $\frac{1}{\sqrt{2\pi}} \cdot \int_k^{\infty} e^{-\frac{\lambda^2}{2}} d\lambda$

C. Sampled data values for various skywave conditions

The tables on the following two pages will show samples taken at 97.5 μs and 102.5 μs after the start of a (non-modulated) Loran-C burst that was filtered with our standard 5th order Butterworth filter ($f_c = 100$ kHz, $B = 20$ kHz). These samples are contaminated with skywaves of various delays and various relative strengths (with respect to the groundwave). Samples taken under skywave conditions that can *not* be encountered within a range of 600 km around a transmitter, are shaded. The unshaded samples show that under any skywave condition, samples taken at 97.5 μs and 102.5 μs always have the same sign, and that the *absolute* value is always larger than 0.748 (top of the envelope of the unfiltered burst is assumed 1). The conclusion that can be drawn is that skywaves within the area of 600 km around a transmitter do not harm Eurofix data detection. For clearness' sake the tables are listed next to each other, so the remainder of this page is left blank intentionally.....

	0 dB	1 dB	2 dB	3 dB	4 dB	5 dB	6 dB	7 dB	8 dB	9 dB	10 dB
55 μs	-0.770 0.821	-0.766 0.814	-0.762 0.806	-0.757 0.798	-0.752 0.788	-0.745 0.777	-0.739 0.765	-0.731 0.751	-0.722 0.736	-0.712 0.718	-0.701 0.699
56 μs	-0.785 0.844	-0.783 0.840	-0.781 0.835	-0.778 0.831	-0.775 0.825	-0.772 0.819	-0.768 0.812	-0.764 0.804	-0.759 0.795	-0.754 0.784	-0.748 0.773
57 μs	-0.803 0.875	-0.803 0.875	-0.803 0.875	-0.804 0.875	-0.804 0.874	-0.804 0.874	-0.804 0.874	-0.804 0.873	-0.805 0.873	-0.805 0.872	-0.805 0.872
58 μs	-0.817 0.901	-0.819 0.904	-0.821 0.907	-0.823 0.911	-0.825 0.915	-0.828 0.920	-0.831 0.925	-0.835 0.931	-0.839 0.938	-0.843 0.945	-0.848 0.953
59 μs	-0.822 0.913	-0.825 0.917	-0.827 0.922	-0.830 0.928	-0.834 0.934	-0.837 0.941	-0.842 0.949	-0.847 0.957	-0.852 0.967	-0.858 0.978	-0.865 0.993
60 μs	-0.819 0.909	-0.821 0.913	-0.823 0.917	-0.826 0.922	-0.828 0.928	-0.832 0.934	-0.835 0.941	-0.839 0.949	-0.844 0.957	-0.849 0.967	-0.854 0.978
61 μs	-0.810 0.894	-0.811 0.896	-0.812 0.899	-0.813 0.901	-0.815 0.904	-0.816 0.908	-0.818 0.911	-0.820 0.915	-0.822 0.920	-0.824 0.925	-0.827 0.931
62 μs	-0.801 0.876	-0.801 0.876	-0.800 0.876	-0.800 0.876	-0.800 0.876	-0.800 0.875	-0.799 0.875	-0.799 0.875	-0.798 0.875	-0.798 0.875	-0.797 0.874
63 μs	-0.794 0.863	-0.793 0.861	-0.792 0.859	-0.791 0.856	-0.790 0.854	-0.788 0.851	-0.786 0.848	-0.785 0.844	-0.782 0.840	-0.780 0.836	-0.777 0.831
64 μs	-0.793 0.857	-0.791 0.855	-0.790 0.852	-0.789 0.849	-0.787 0.846	-0.785 0.842	-0.783 0.838	-0.781 0.833	-0.778 0.827	-0.775 0.821	-0.772 0.815
65 μs	-0.795 0.861	-0.794 0.859	-0.793 0.856	-0.792 0.854	-0.791 0.851	-0.789 0.848	-0.788 0.844	-0.786 0.840	-0.784 0.836	-0.782 0.831	-0.779 0.825
66 μs	-0.799 0.869	-0.799 0.868	-0.799 0.867	-0.798 0.866	-0.798 0.865	-0.797 0.863	-0.797 0.862	-0.796 0.860	-0.795 0.858	-0.794 0.855	-0.793 0.853
67 μs	-0.804 0.879	-0.804 0.879	-0.804 0.879	-0.804 0.879	-0.805 0.880	-0.805 0.880	-0.805 0.880	-0.805 0.881	-0.806 0.881	-0.806 0.882	-0.807 0.882
68 μs	-0.806 0.885	-0.807 0.886	-0.807 0.887	-0.808 0.888	-0.808 0.890	-0.809 0.891	-0.810 0.893	-0.811 0.895	-0.812 0.897	-0.813 0.900	-0.814 0.902
69 μs	-0.806 0.887	-0.807 0.888	-0.808 0.889	-0.808 0.891	-0.809 0.892	-0.810 0.894	-0.811 0.896	-0.812 0.899	-0.813 0.901	-0.814 0.904	-0.815 0.908
70 μs	-0.805 0.884	-0.805 0.885	-0.806 0.886	-0.806 0.887	-0.807 0.889	-0.807 0.890	-0.808 0.892	-0.808 0.894	-0.809 0.896	-0.810 0.898	-0.811 0.900
71 μs	-0.803 0.880	-0.803 0.880	-0.803 0.881	-0.803 0.881	-0.804 0.882	-0.804 0.882	-0.804 0.883	-0.804 0.884	-0.804 0.884	-0.805 0.885	-0.805 0.886
72 μs	-0.802 0.876	-0.801 0.876	-0.801 0.875	-0.801 0.875	-0.801 0.875	-0.801 0.875	-0.801 0.874	-0.801 0.874	-0.800 0.874	-0.800 0.873	-0.800 0.873
73 μs	-0.801 0.873	-0.801 0.873	-0.800 0.872	-0.800 0.872	-0.800 0.871	-0.800 0.870	-0.799 0.869	-0.799 0.869	-0.798 0.867	-0.798 0.866	-0.797 0.865
74 μs	-0.801 0.873	-0.801 0.872	-0.801 0.872	-0.800 0.871	-0.800 0.871	-0.800 0.870	-0.800 0.869	-0.799 0.868	-0.799 0.867	-0.798 0.866	-0.798 0.864
75 μs	-0.802 0.874	-0.801 0.874	-0.801 0.874	-0.801 0.873	-0.801 0.873	-0.801 0.872	-0.801 0.872	-0.801 0.871	-0.800 0.870	-0.800 0.869	-0.800 0.866
76 μs	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.875
77 μs	-0.803 0.878	-0.803 0.878	-0.803 0.878	-0.803 0.878	-0.803 0.878	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.880
78 μs	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.880	-0.803 0.880	-0.803 0.880	-0.803 0.881	-0.804 0.881	-0.804 0.881	-0.804 0.882
79 μs	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.880	-0.803 0.880	-0.803 0.880	-0.803 0.881	-0.803 0.881	-0.803 0.882
80 μs	-0.803 0.878	-0.803 0.878	-0.803 0.878	-0.803 0.878	-0.803 0.878	-0.803 0.878	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.879	-0.803 0.880
81 μs	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877
82 μs	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876
83 μs	-0.802 0.877	-0.802 0.877	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876
84 μs	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.876
85 μs	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877	-0.802 0.877

	11 dB	12 dB	13 dB	14 dB	15 dB	16 dB	17 dB	18 dB	19 dB	20 dB	21 dB
55 μ s	-0.689 0.677	-0.675 0.653	-0.659 0.625	-0.642 0.595	-0.622 0.560	-0.600 0.522	-0.576 0.478	-0.548 0.430	-0.517 0.375	-0.482 0.314	-0.443 0.245
56 μ s	-0.742 0.760	-0.734 0.746	-0.726 0.730	-0.717 0.712	-0.706 0.692	-0.694 0.670	-0.681 0.644	-0.666 0.616	-0.650 0.584	-0.631 0.548	-0.610 0.508
57 μ s	-0.806 0.871	-0.806 0.870	-0.806 0.870	-0.807 0.869	-0.807 0.868	-0.808 0.867	-0.809 0.865	-0.810 0.864	-0.810 0.862	-0.811 0.860	-0.812 0.858
58 μ s	-0.854 0.963	-0.860 0.973	-0.868 0.985	-0.875 0.998	-0.884 1.013	-0.894 1.029	-0.906 1.048	-0.918 1.069	-0.932 1.092	-0.948 1.118	-0.966 1.148
59 μ s	-0.872 1.004	-0.881 1.020	-0.891 1.037	-0.901 1.057	-0.913 1.079	-0.927 1.104	-0.942 1.131	-0.959 1.162	-0.978 1.197	-1.000 1.236	-1.024 1.280
60 μ s	-0.861 0.991	-0.868 1.004	-0.876 1.020	-0.885 1.037	-0.895 1.057	-0.906 1.079	-0.918 1.104	-0.933 1.131	-0.948 1.162	-0.966 1.197	-0.986 1.236
61 μ s	-0.830 0.938	-0.833 0.945	-0.837 0.954	-0.841 0.963	-0.846 0.973	-0.851 0.985	-0.857 0.998	-0.864 1.013	-0.872 1.030	-0.880 1.048	-0.889 1.069
62 μ s	-0.797 0.874	-0.796 0.874	-0.795 0.873	-0.794 0.873	-0.793 0.872	-0.792 0.871	-0.791 0.871	-0.790 0.870	-0.788 0.869	-0.786 0.868	-0.784 0.867
63 μ s	-0.774 0.825	-0.771 0.819	-0.767 0.812	-0.762 0.804	-0.757 0.795	-0.752 0.785	-0.746 0.774	-0.739 0.761	-0.731 0.747	-0.723 0.731	-0.713 0.713
64 μ s	-0.768 0.807	-0.764 0.798	-0.759 0.789	-0.754 0.778	-0.748 0.766	-0.741 0.752	-0.734 0.737	-0.725 0.720	-0.716 0.701	-0.705 0.680	-0.694 0.655
65 μ s	-0.776 0.819	-0.773 0.812	-0.770 0.804	-0.766 0.795	-0.761 0.785	-0.756 0.774	-0.750 0.761	-0.744 0.747	-0.737 0.731	-0.729 0.713	-0.720 0.693
66 μ s	-0.792 0.850	-0.791 0.846	-0.789 0.842	-0.788 0.838	-0.786 0.833	-0.784 0.828	-0.782 0.822	-0.779 0.815	-0.777 0.808	-0.773 0.799	-0.770 0.790
67 μ s	-0.807 0.883	-0.808 0.883	-0.808 0.884	-0.809 0.885	-0.810 0.886	-0.811 0.887	-0.812 0.888	-0.813 0.890	-0.814 0.891	-0.816 0.893	-0.818 0.895
68 μ s	-0.816 0.905	-0.818 0.909	-0.819 0.913	-0.822 0.917	-0.824 0.922	-0.827 0.927	-0.829 0.934	-0.833 0.940	-0.836 0.948	-0.841 0.957	-0.845 0.967
69 μ s	-0.817 0.912	-0.819 0.916	-0.821 0.920	-0.823 0.926	-0.825 0.932	-0.828 0.938	-0.831 0.946	-0.835 0.954	-0.839 0.964	-0.843 0.974	-0.848 0.986
70 μ s	-0.812 0.903	-0.813 0.906	-0.815 0.910	-0.816 0.914	-0.818 0.918	-0.820 0.923	-0.822 0.929	-0.824 0.935	-0.827 0.943	-0.830 0.950	-0.833 0.959
71 μ s	-0.805 0.887	-0.805 0.889	-0.806 0.890	-0.806 0.892	-0.807 0.893	-0.807 0.895	-0.808 0.898	-0.809 0.900	-0.809 0.903	-0.810 0.906	-0.811 0.910
72 μ s	-0.799 0.872	-0.799 0.872	-0.799 0.871	-0.798 0.870	-0.798 0.869	-0.797 0.869	-0.797 0.868	-0.796 0.866	-0.795 0.865	-0.794 0.864	-0.793 0.862
73 μ s	-0.797 0.864	-0.796 0.862	-0.795 0.860	-0.795 0.858	-0.794 0.856	-0.793 0.853	-0.791 0.850	-0.790 0.847	-0.789 0.843	-0.787 0.839	-0.785 0.834
74 μ s	-0.797 0.863	-0.797 0.861	-0.796 0.859	-0.795 0.857	-0.794 0.854	-0.793 0.851	-0.792 0.848	-0.791 0.844	-0.790 0.841	-0.788 0.836	-0.787 0.831
75 μ s	-0.800 0.867	-0.799 0.866	-0.799 0.865	-0.798 0.863	-0.798 0.862	-0.797 0.860	-0.797 0.858	-0.796 0.855	-0.795 0.853	-0.794 0.850	-0.794 0.846
76 μ s	-0.802 0.874	-0.802 0.874	-0.802 0.874	-0.802 0.873	-0.802 0.873	-0.802 0.872	-0.802 0.872	-0.802 0.871	-0.802 0.870	-0.801 0.869	-0.801 0.868
77 μ s	-0.804 0.880	-0.804 0.880	-0.804 0.881	-0.804 0.881	-0.804 0.882	-0.805 0.882	-0.805 0.883	-0.805 0.884	-0.806 0.884	-0.806 0.885	-0.806 0.886
78 μ s	-0.804 0.883	-0.804 0.883	-0.805 0.884	-0.805 0.885	-0.805 0.886	-0.805 0.887	-0.806 0.888	-0.806 0.889	-0.807 0.891	-0.807 0.892	-0.808 0.894
79 μ s	-0.804 0.882	-0.804 0.883	-0.804 0.883	-0.804 0.884	-0.804 0.885	-0.805 0.886	-0.805 0.887	-0.805 0.888	-0.806 0.890	-0.806 0.891	-0.806 0.893
80 μ s	-0.803 0.880	-0.803 0.880	-0.803 0.881	-0.803 0.881	-0.803 0.882	-0.803 0.882	-0.803 0.883	-0.804 0.883	-0.804 0.884	-0.804 0.885	-0.804 0.886
81 μ s	-0.802 0.877	-0.802 0.877	-0.802 0.878	-0.802 0.878	-0.802 0.878	-0.802 0.878	-0.802 0.878	-0.802 0.878	-0.802 0.878	-0.802 0.878	-0.802 0.878
82 μ s	-0.802 0.876	-0.802 0.876	-0.802 0.876	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.874	-0.801 0.874	-0.801 0.874	-0.801 0.873
83 μ s	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.874	-0.802 0.874	-0.802 0.874	-0.802 0.873	-0.801 0.873	-0.801 0.872	-0.801 0.872
84 μ s	-0.802 0.876	-0.802 0.876	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.874	-0.802 0.874	-0.802 0.874	-0.802 0.873
85 μ s	-0.802 0.877	-0.802 0.877	-0.802 0.876	-0.802 0.876	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.875	-0.802 0.875

DELFT UNIVERSITY OF TECHNOLOGY
Faculty of Electrical Engineering
Telecommunication and Traffic
Control Systems Group

Simulating the Eurofix Data Link

A.M. Noorbergen

Date: June 1993

Contents: This report describes the modifications that were made in a Dynamic Dual-Chain Loran-C Simulator. These modifications allow the simulator to apply Eurofix phase modulation to the generated Loran-C signal.

Professor: D. van Willigen
Mentors: M. Braasch, E.J. Breeuwer, D.J.R. van Nee
Assignmentnumber: A-458
Period: August 1992 - June 1993

After the introduction of the Eurofix concept for transmission of Differential GPS data via Loran-C signal, the structure and the internals of a Dynamic Dual-Chain Loran-C Simulator are explained. This Loran-C Simulator was developed in 1987 at the Delft University of Technology. This simulator has been modified to simulate Eurofix signals, which are Loran-C signals that can be phase modulated. These modifications are described in this report.

Indexing terms: Radio Navigation, Phase Modulation, Coding, Radio Propagation, Integrated Navigation, Loran-C, DGPS, Eurofix

Summary

Eurofix is a combination of Differential GPS and Loran-C, in which DGPS corrections are transmitted to a user via additional phase modulation of the Loran-C signal. The nature of the Loran-C signal structure only allows for very small bit rates. Despite these small bit rates, position accuracies of 8 to 20 meter in the horizontal plane (95 %) can be achieved by utilising atomic frequency references that are already present in Loran-C transmitters. The specifications, set in the Federal Radio navigation Plan for Harbour Approaches (1992) can be met. Using Eurofix, the user has a low-cost, accurate positioning system at his disposal with an improved integrity and availability, compared to standard GPS or standard Loran-C.

To investigate our ideas on Eurofix, and to perform actual measurements on our Eurofix system, the need was felt to have a simulator at our disposal that could generate phase modulated Loran-C signals. In 1987, a Dynamic Dual-Chain Loran-C Simulator was developed at the Delft University of Technology, that could (real-time) generate Loran-C signals. This signal generator can be controlled by means of a Personal Computer. It was decided to modify this simulator to enable Eurofix phase modulation.

This report explains the internals of the Loran-C simulator mentioned above. The Loran-C simulator mainly consists of a signal generator that can generate Loran-C bursts. The timing of the simulator is controlled by a Z-80 microprocessor. Eurofix modulations is a matter of generating bursts at alternating moments in time, so modifications for Eurofix turned out to be a matter of software only. The modified software is downwards compatible with the non-Eurofix version: all PC-control software that has been written for the "old" simulator can still be used. The modifications that were necessary are explained in the remainder of this report

Table of contents

1.	Introduction	5
2.	Introduction to Eurofix	7
2.1	Differential GPS	7
2.2	The Eurofix solution	7
2.3	The Eurofix datalink	8
3.	The Dynamic Dual-Chain Loran-C Simulator	11
3.1	Features of the Loran-C Simulator	11
3.2	The Loran-C Simulator Hardware	11
3.2.1	The Loran-C burst generator	12
3.2.2	A sinewave generator with microprocessor controlable phase	12
3.2.3	Blockdiagram of the Loran-C Simulator	13
3.2.4	The microprocessor unit	14
3.3	Software of the Loran-C Simulator	14
3.3.1	Communication protocol between Simulator and outside world	14
3.3.2	Control commands from outside world	15
3.3.3	Communication module (SIMIO.MAC)	15
3.3.4	Main module (SIM.MAC)	16
3.3.5	Burst Processing module (SIMBRS.MAC)	17
3.3.6	Test module (SIMTEST.MAC)	19
4.	Modification of the Loran-C simulator for data transmission	21
4.1	Introduction	21
4.2	Design considerations for the Eurofix simulator modifications	22
4.3	Two major modifications in the Loran-C simulator	23
4.3.1	Buffering of databits	23
4.3.2	Setting timer and delay lines for Eurofix modulation	24
4.4	Changes in the original software	25
4.4.1	Communication module (SIMIO.MAC)	26
4.4.2	Main module (SIM.MAC)	26
4.4.3	Burst processing module (SIMBRS.MAC)	26
4.5	Performance analysis of the modified Loran-C simulator	27
4.6	Modifications in a nutshell	29
APPENDIX A: Operations manual Dynamic Dual-Chain Loran-C Simulator		31
APPENDIX B: Source code listings of original software		39
APPENDIX C: Source code listings of modified software		83
Literature references		131

1. Introduction

Integrated navigation is a topic of great interest at the moment and the years to come. International borders disappear at a quick rate and people have become more mobile. This generates larger flows of traffic on the already overcrowded infrastructures. The growing traffic load can be led into channels by utilising the achievements of modern technology in the field of radio navigation systems.

Safety and integrity play an important role in the development of (new) radio navigation systems: people entrust their lives to those systems. That is why demands on radio navigation systems have become more stringent. Authorities oblige large vessels for instance to have at least two independent navigation systems aboard. Integration of radio navigation systems opens ways to a larger accuracy and a larger integrity by combining the strong points of 2 or more systems.

At the Delft University of Technology, department of Electrical Engineering, research is done on the integration of Loran-C and GPS under the name EUROFIX. In the Eurofix proposal the Loran-C system is not only used for navigation, but also for data transmission. This data transmission feature will be used to transmit DGPS data from a DGPS reference station to a mobile user. With a slightly modified Loran-C receiver, the user will have DGPS information at his disposal, which offers him a greater accuracy (10 to 20 meter) and a greater integrity. In this way, DGPS corrections are available in a large part of Europe at minimal user cost.

A preliminary Eurofix system has already been proposed at several occasions. But because there are a lot of new transmitters planned in Europe, it is important to convince the authorities to consider to implement Eurofix in these new transmitters. For that reason some haste was imperative, and just for that reason it was decided to create a Eurofix test set-up. One part of my graduation work was to modify a Dynamic Dual-Chain Loran-C simulator, to make it possible to transmit Eurofix data in its preliminary form. Together with a modified receiver it is then possible to show that Eurofix can work basically.

The datalink that is constituted in the Eurofix proposal hasn't been the subject of a thorough investigation. It is merely based on the ideas of the Clarinet Pilgrim system of the US Navy, in which the 6 last burst of a group of eight of a station are somewhat shifted with respect to each other. Using different shift patterns, data can be transmitted. The other part of my graduation work was to investigate this datalink and to find an answer on the following questions: can the modulation principle be improved, what data-rates can be achieved, what bit-error rates can we expect and what can we do about bit errors, and how will Eurofix affect conventional users?

The report in front of you describes the first part of my graduation work: the modification of a Dynamic Dual-Chain Loran-C simulator to add data transmission capabilities.

2. Introduction to Eurofix

As mentioned in the introduction, Eurofix is a combination of Loran-C and GPS. Why this combination offers additional value to a user, and how this is done, will be explained in this section.

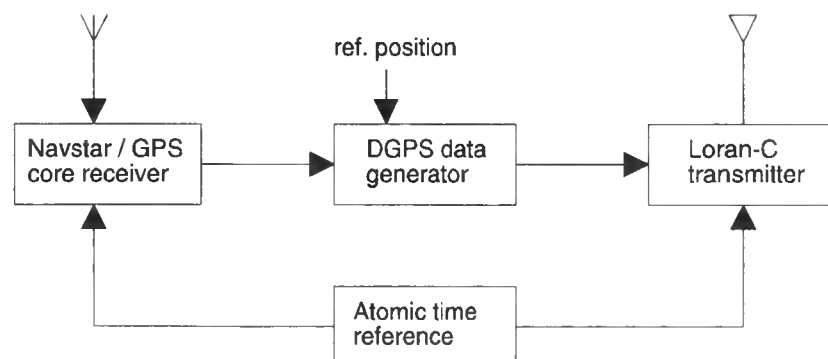
2.1 Differential GPS

The Global Positioning System (GPS) is a satellite-based radio navigation and positioning system, that is being operated by the United States Department of Defence. This systems offers authorised users accuracies of 10 to 20 meter (95 percentile level). Civilian users will have access to a less accurate system: 100 meters. This worse accuracy is due to the deliberate disturbances, called Selective Availability, that the US Department of Defence introduces into the "civilian signal".

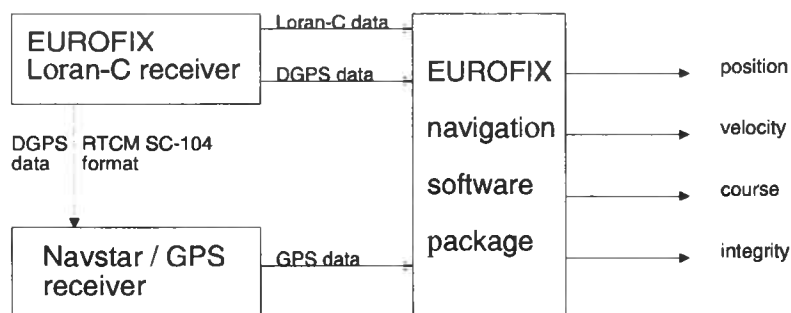
However, these disturbances can be diminished for a great deal with Differential GPS: a GPS receiver at an exactly known location can calculate the error that is introduced by S.A. When a mobile user compensates his own GPS-position for this error, he will have a better accuracy. The ultimate mobile user DGPS accuracy depends on the age of the compensation message, because S.A. varies in time and on distance to the reference station (because the common S.A. error decorrelates in space). The only problem is: how to supply a mobile user this correction information? For this, a data communication link is needed. The International Association of Lighthouse Authorities has proposed to let radio beacons transmit DGPS data. In Eurofix, Loran-C can constitute such a data link. Although nothing has been decided yet on the physical data link, the data that should be transmitted is already standardised by the Radio Technical Commission for Maritime Services (RTCM). This DGPS standard is known as SC-104.

2.2 The Eurofix solution

The Eurofix concept integrates Navstar/GPS with Loran-C. Differential GPS corrections are phase coded onto the Loran-C signals. This can be done quite easy and cheap. The data rates that can be established vary from 10 to 25 bits per second, however, Beekhuis [8] showed that, even with such a low data rate, a very good DGPS performance can be achieved (better than the specifications of the Federal Radio navigation Plan for Harbour Approaches), a the cost of using atomic references.



From a safety point of view, a lot of users might want to have both a GPS- and a Loran-C receiver at their disposal. They only need a slightly modified Loran-C receiver to have DGPS corrections at their disposal as well. The Eurofix receiver set up can be described on the basis of the following figure :

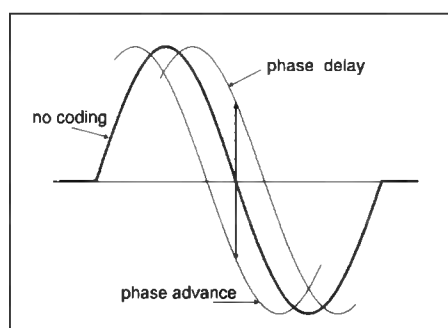


The Eurofix Loran-C receiver generates Loran positions. It also decodes the DGPS data from the Loran-C signal. This DGPS data might be available at an RTCM SC-104 compatible output. The DGPS output could also be connected to some "navigation package", that could take care of integrity monitoring and hybridisation.

This proposed cost-effective Eurofix navigation system offers increased accuracy and integrity over either GPS or Loran-C separately, without the need for a separate data channel that uses additional bandwidth in the scarce radio spectrum.

2.3 The Eurofix data link

In Eurofix, such the data link is constituted by modulation of the Loran-C signal. In the original Eurofix proposals, the last six bursts of the group of eight bursts are shifted with respect to each other, in the following way:



Binary data bit value	Modulation pattern							
1	0	0	+	-	+	-	+	-
0	0	0	-	+	-	+	-	+

Modulation pattern for code rate of 6

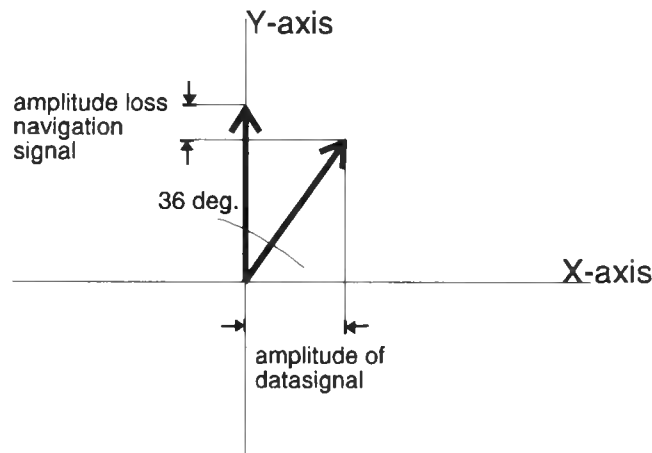
0 = no time shift

+ = positive code bit = 1 μ s phase delay

- = negative code bit = 1 μ s phase advance

Burst 3..8 of each transmitter are shifted $1\ \mu\text{s}$. Either they come $1\ \mu\text{s}$ earlier, or they will be delayed $1\ \mu\text{s}$. In the table above, a $1\ \mu\text{s}$ early burst is denoted by a -, a burst that is delayed $1\ \mu\text{s}$ is denoted by a + character. Data bits can be transmitted by applying different modulation patterns to the last 6 bursts. In this way it is possible to transmit 1 bit per station per GRI. This limits the data rate 10 to 25 bits per second.

Conventional Loran-C receivers will not notice this modulation, as long as the number of burst advances equals the number of burst delays. The modulation effects are averaged out in this way, assuming that the lowest modulation frequency is above the bandwidth of the tracking loop of the receiver. However, the conventional user will notice a slightly deteriorated Signal-to-Noise ratio of 1.34 dB. I will explain that using the following phasor diagram:



The conventional user will expect the phasor that is standing upright. Due to the $1\ \mu\text{s}$ shift (which corresponds to 36°), the user will find a slightly rotated phasor. Averaging 6 of those rotated phasor gives the same zero-crossings, but because of the phase shift, the amplitude of the averaged bursts will be less than normal (the distance at the Y-axis). As you can see, a smaller modulation index would cause a smaller loss in the amplitude of the "navigation" component, but it also causes a smaller amplitude of the quadrature component: the DGPS data signal.

Somewhere an optimum between power loss of the conventional Loran-C user and the signal to noise ratio of the DGPS data signal will have to be found. This topic is still under research.

3. The Dynamic Dual-Chain Loran-C Simulator

This chapter will give you an insight in the internals of the Dynamic Dual-Chain Loran-C Simulator that was developed at the Delft University of Technology, in 1987. At that time the need was felt to have a simulator available with which it was possible to investigate dynamic behaviour of Loran-C receivers. Of course it is impractical to travel the route you wish to investigate, and sometimes it is even impossible to carry out specific moves (for instance step response). The design and development of this simulator was a Master's Project of René Kellenbach at the department of Electronic Technology (Ned: Elektronische Techniek). The simulator is described in details in his thesis [1].

3.1 Features of the Loran-C Simulator

The design demands the simulator would have to meet (at that time) were:

- Generation of Loran-C bursts according to Coast Guard specifications, cat. 1 transmitters.
- Computer controllable via an RS-232 interface.
- Real-Time simulation with freely programmable td's via computer, resolution of 25 ns.
- GRI and number of transmitters in chain programmable
- Programmable Skywaves (delay and strength) according to Coast Guard Minimum Performance Standards
- Programmable noise generator
- 50 Ω output for receiver
- Internal clock generator as well as input for external 5 MHz Cesium standard
- Dynamic programmable Envelope to Carrier Delay, Signal to Noise Ratio and skywave delay, programmable for each transmitter independently
- Dual chain
- Very good phase stability (to allow rho-rho navigation)

The simulator that was develop does meet all those requirements. To meet those requirements, a microprocessor had to be brought into action for controlling such the complex task of generating Loran-C bursts dynamically. The hardware of the simulator will be discussed first. The software will be treated separately.

3.2 The Loran-C Simulator hardware

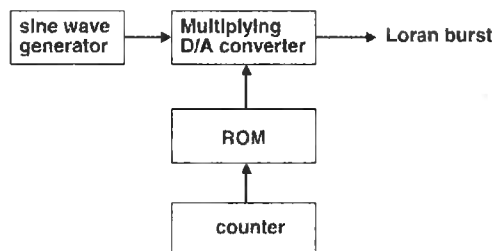
Fortunately, the Loran-C simulator is in fact no more than a clock and timing unit, a signal generator that generates Loran C bursts at the command of a microprocessor and a microprocessor system. This means that all "intelligence" of the simulator can be put in EPROM, which enables us to make a lot of changes without having to modify the hardware. For us, the most important and interesting parts of the hardware are the burst generator, the carrier generator and the microprocessor system.

3.2.1 The Loran-C burst generator

Loran-C bursts can be described mathematically by

$$e(t) = A \cdot \left(\frac{t}{65}\right)^2 \cdot \exp\left(\frac{2-2 \cdot t}{65}\right) \cdot \cos(\omega \cdot t + PC)$$

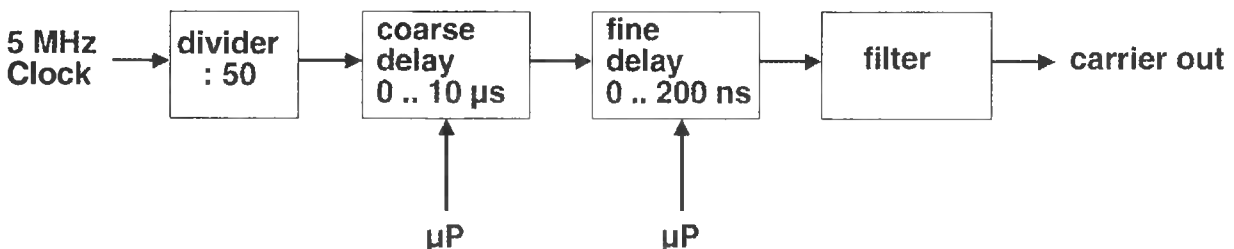
in which $e(t)$ stands for strength of the electrical field, A for a constant amplitude, ω for the carrier frequency and PC for the phase code of the signal, which can be 0 or π . Generating such a burst is just a matter of multiplying a sine wave carrier with an envelope. The form of the envelope is defined by the second and third factor of above equation. The signal generator that generates the bursts in the simulator can be described with the following block diagram:



The sine wave generator generates a 100 KHz sine wave. The multiplying D/A converter modulates that wave into a Loran-C burst by multiplying that carrier with the Loran-C envelope. The samples of the shape of the envelope, which depend on the ECD you want to simulate, are stored in Read Only Memory. By choosing the appropriate bank in ROM, you can get different shapes (ECD's). The phase code of the Loran burst can be set by multiplying the burst by 1 or -1.

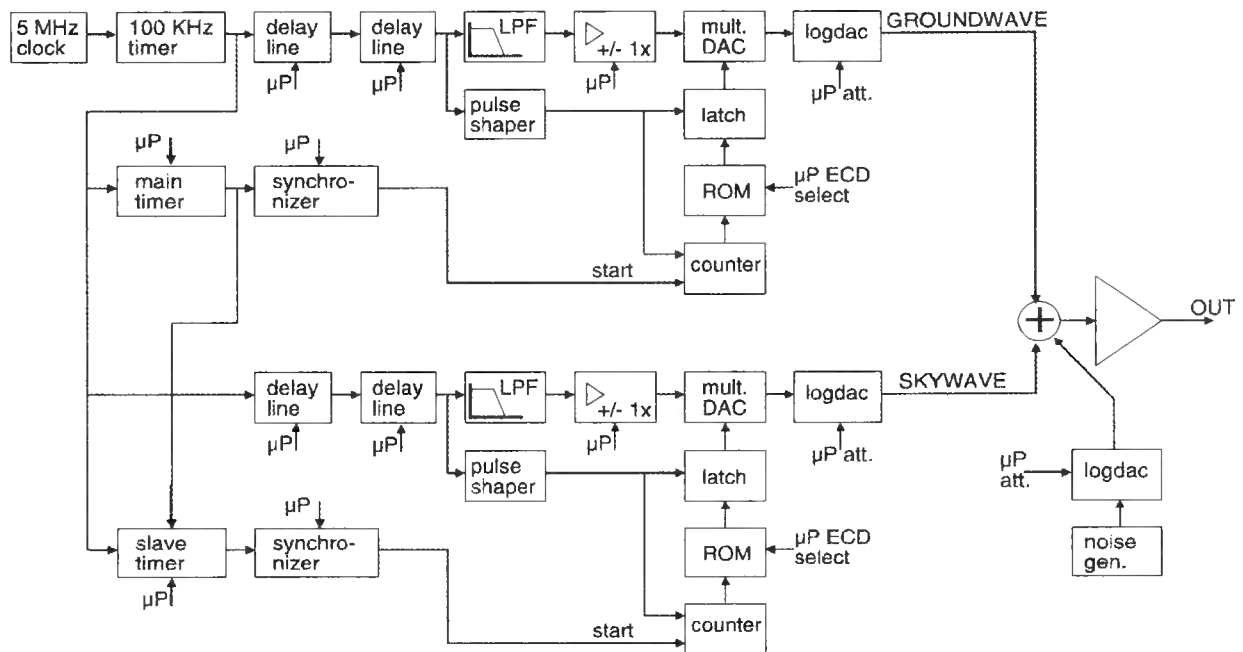
3.2.2 A sine wave generator with microprocessor-controllable phase

The 5 MHz clock that is present in the simulator, is divided by 50 to generate a 100 KHz square wave. The resulting square wave is fed through a low-pass filter, which yields a 100 KHz carrier sine wave. The phase of the (digital) square wave can be easily controlled by a microprocessor programmable delay line. As you can see in the following block diagram, the delay line is split up into a part that can delay the signal 10 μ s (one complete cycle) with a resolution of 200 ns (and a part that can delay the signal 200 ns at most (with a resolution of 25 ns). The coarse delay line consists of a counter and a comparator. The counter counts from 0 to 49 with a programmable preset. By changing the preset, we can shift the phase of the counter. Using the comparator it is possible to generate a symmetrical square wave.



3.2.3 Block diagram of the Loran-C Simulator

The carrier generator with programmable phase and the burst generator with envelope multiplier are the two main components on which the following block diagram of the simulator is based:



As you may have noticed: the block diagram consists of two almost identical parts. The upper part generates the ground wave, whereas the lower part generates the skywave. These two signals are added, together with noise, and amplified.

At the upper left top there is the 5 MHz clock which is divided by 50 with the help of a programmable timer. This 100 KHz square wave is used as input for the delay lines, to create a 100 KHz sine wave. The 100 KHz square wave is also fed into the main- and the slave timer. The main timer generates startpulses for the pulse generator of the ground waves; the slave timer generates startpulses for the pulse generator of the skywaves. This slave timer is a programmable one-shot timer that is controlled by the main timer. This allows us to program the delay of the skywave. From the two delay lines, the square wave is led to a low-pass filter to yield a carrier sine. After a programmable inverter, this carrier will be multiplied with the Loran-C envelope in the multiplying DAC. The envelope of the Loran-C burst is stored in ROM. By selecting different blocks in this ROM, different ECD's can be selected. The resulting Loran-C burst can be attenuated by a programmable logarithmic attenuator (LOGDAC). The skywave is generated by a similar generator and added to the ground wave. Finally, noise from a pseudo-random noise generator is added.

As you can see, the microprocessor can control ground- and skywave phase, ground- and skywave polarity, ground- and skywave level (using the LOGDAC's), ground- and skywave ECD, skywave delay and noise power.

To provide Dual-Chain capabilities, the simulator is equipped with two generators as sketched above. They both have their own microprocessor unit, so they can operate independently. The only thing they do share is an RS-232 communication line to a controlling Personal Computer.

3.2.4 The microprocessor unit

The microprocessor part of each chain consists of the famous Zilog Z80 microprocessor, which runs at a clock frequency of 5 MHz. It consist furthermore of 8 Kilobytes of EPROM memory, 8 Kilobytes of RAM memory and a few I/O ports, through which the signal generators are controlled. Each microprocessor part also has an RS-232 interface, based on an INTEL 8251 Programmable Communication Controller. The microprocessor board is equipped with 8 dip-switches. At start-up, the microprocessor will take a look at these switches and decides whether to run a test program, or to go into normal operation.

One of the switches is used to set the address of the simulator. The two interfaces of each chain are switched in parallel. Each chain has its own select code, determined by the setting of the dipswitch (see Appendix A), so each chain can make up for whom the commands from the host are destined. The chain that is currently selected is the only chain that is allowed to *send* data back to the host.

The programmable *timer* that has been mentioned is the (again) famous 8253. Datasheets of this chip, and of the 8251, can be found in appendix D, so that you know how to program them.

3.3 Software of the Loran-C Simulator

A great deal of the effort of the development of the simulator has been put in the development of the *software*. Fortunately, much of the control of the simulator has been left over to software, which makes future changes relatively easy. Controlling the simulator as a whole is also a matter of software: you won't find any switches, buttons, keys or knobs on the simulator, except for the On/Off switch. the operation of the simulator is completely controlled by commands that can be fed into the simulator via an RS-232 interface. In the rest of this report, it is assumed that a Personal Computer is connected to the other side of the RS-232 cable. The software can be divided up into several modules. But before discussing each module, I would like to spend a few words on the communication between simulator and Personal Computer. The source code of the software can be found in appendix B.

3.3.1 Communication protocol between simulator and outside world

The simulator uses an advanced protocol for its communication with the PC. Under all circumstances we must avoid that the not-selected chain A (that also scans the line for its "select" command) thinks it is selected when listening to data destined for chain B.

All binary data will be converted into readable ASCII characters (codes 20 .. 7F hex). Every parameter or command is at least one byte long, with the following structure:

0 1 s d d d d (= ASCII 20 .. 3F hex)

In this format, *s* is the signbit (1 = pos, 0 = neg) and *d d d d* is a 4-bit datafield.. Using this one byte we can send parameters varying from -15 to + 15. If we want to transmit larger values, we let this <low> byte precede by one or more <high> bytes, with the following format:

1 *d d d d d d* (= ASCII 40 .. 7F hex)

Now we have a six-bit datafield *d d d d d d*. Using a <low> and a <high> byte we can send 10 bits numbers; 16 bit numbers can be send with another <high> byte. One of the appendices will provide you high-level PC software that can do the conversion from "normal" numbers to numbers in this protocol.

The ASCII codes 00 .. 1F aren't used in this protocol, so they can be used for lead-in sequences. You can find a complete list of commands for the control of the simulator and their format in appendix A.

3.3.2 Control commands from the outside world

The PC controlling the simulator can send various commands to the simulator. These vary from setting the TD's to setting the output level of the noise generator. As said before, a complete list of available commands can be found in appendix A.

Some commands will be carried out directly after reception. Most commands however will be placed into the command buffer. The simulator will carry out these commands in order of reception. If the buffer runs out of commands, the simulator will continue to generate signals according to the last setting, i.e. the simulator will "stand still" in this case. The buffer has room for about 6000 bytes.

The software of the simulator is divided into 4 modules: SIM.MAC, SIMIO.MAC, SIMBRS.MAC and SIMTEST.MAC. The source code of these modules can be found in appendix B.

3.3.3 Communication module (SIMIO.MAC)

The file SIMIO.MAC contains source code for all communication with the Personal Computer. The serial interface chip (8251A) will have to be initialised. The function "INITRS" will take care of that. This routine will set the data format (8 bits, no parity, 2 stop bits). The pointers for the command buffer (RDPNT and WRPNT) will also be initialised, as well as the counter that counts the buffer contents (BUFCNT). The dipswitch that tells the simulator to which select code to listen will also be read in this routine.

When the 8251A has received a command from the RS-232 connection to the PC, it generates an interrupt request for the Z80 microprocessor. The Z80 calls the interrupt service handler, in which the character will be processed. If it is a SELECT command, the simulator will be selected if the code corresponds to the dipswitch setting and "SELFLG" will be set, or else it will deselect itself. In case of another command, the interrupt handler routine will check whether the simulator is selected. If this is the case, the command will be processed. If not, it will discard the character.

Commands will be split into direct commands and buffered commands. Direct commands will be carried out immediately, buffered commands will be placed in the buffer

("BUFFER"). "WRPNT" will keep track of the last position written to, and "RDPNT" will keep track of the next character to be read from the command buffer. If one of these pointers reaches the end of the buffer, it will be reset to the begin of the buffer to simulate a FIFO buffer. If the PC issues a request for the status of the buffer, the interrupt service routine will use the routine "SEND" to send the number of bytes in the buffer to the PC.

The main program can get characters from the buffer using the function GET_BYT. For this purpose there will be first a check (by means of RX_STAT) whether there are characters in the buffer. If this is not the case, the routine will wait until there are. The routine BUF_READ is used to fetch the character from the buffer and to adjust the pointers.

3.3.4 Main module (SIM.MAC)

We now know how to communicate with the PC; the working of the main program can be made clear now. After a reset at power-up, the Z80 initialises the RS-232 interface (INITRS), the burst processing routines (INITBRS) and starts the noise generator. It will then take a look at the dipswitch to see if a test program should be run.

At this point, the simulator is ready to process commands from the buffer. During processing of these commands, interrupts may occur from both the serial interface (character received) and the main timer (timer 1). In the INTERRUPT routine the status register of the 8251 serial communication controller will be read. This tells us the cause of the interrupt. The appropriate interrupt handler (RX_INT or BRS_INT) will be called.

The main program is a simple endless loop (GET_CMD), in which the commands are taken out of the buffer one at a time and are decoded. After reception of a GS-code (= relative TD move, see Appendix A), GET_CMD will jump to the corresponding routine (REL_MOVE) immediately. A received ESCAPE code will be decoded by means of a look-up table, in which the addresses for several routines can be found.

All routines that need parameters, use a special routine to get those parameters out of the buffer: GET_INT. In this routine, the protocol discussed on page 14 is implemented. It will convert the bytes received from the PC into ordinary binary numbers of at most 32 bits, sufficient for the longest parameter we can expect (22 bits for a DTA).

The REL_MOVE routine, with which the dynamic simulation by means of relative moves is controlled, initialises a data-table (MOV_TIM) in which the moves to make are stored. The burst interrupt processing routines (more on that later), detect this and will carry out these relative moves real-time. The routines are mutually synchronised by a semaphore table (MOV_FLG).

The other routines also use tables in which parameters are stored. These parameters will be used to adjust the simulator the next GRI. This is a cyclic process: when all eight bursts of a station have been generated, the burst interrupt software initialises the burstgenerators for the next transmitter in the chain. The data needed for this is fetched from a table using indexing.

Because all routines are more or less the same qua structure, a table with some names and corresponding routines will hopefully suffice:

function	routine	table
DTA initialisation	SET_DTA	DTATAB
set skywave delay	SET_SDEL	SKYTAB
set ground wave delay	SET_GLEV	GND_LEV
set skywave delay	SET_SLEV	SKY_LEV
set ground wave ECD	SET_GECD	G_ECDTB
set skywave ECD	SET_SECD	S_ECDTB

3.3.5 Burst processing module (SIMBRS.MAC)

The job of the simulator is to generate Loran-C bursts as specified by a Personal Computer. This job is really done in this module. The module SIMIO.MAC only provides a communication link and SIM.MAC only translates commands from the PC and places these parameters in tables. SIMBRS.MAC is going to process these tables and control the signal generators from these tables. The complete process of generating bursts is under strict control of the main timer (timer 1 of a 8253 chip). This timer operates on a 100 KHz clock frequency and has a range of 16 bits. We can define time intervals up to $65535 * 10 \mu s = 0.65 s$, more than sufficient to generate all TD's.

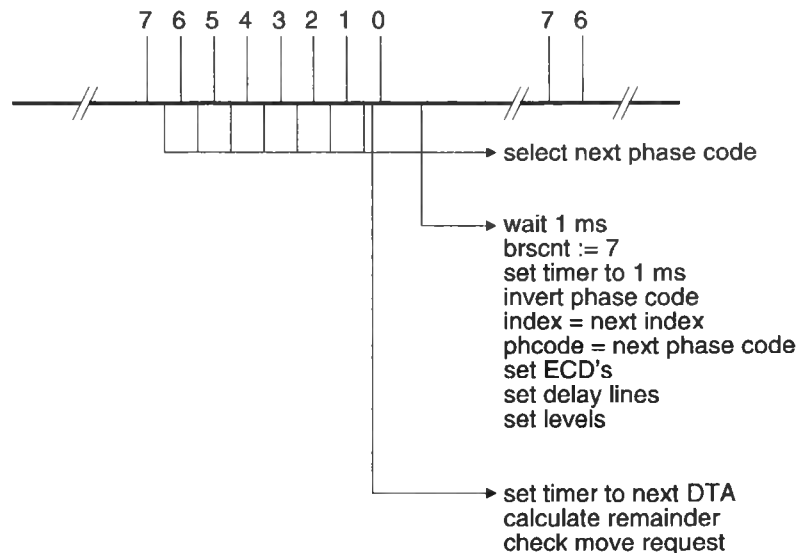
When this timer is loaded with a number, it starts counting down to 0. When the timer reaches 0, it generates an interrupt for the Z80, which should take action on this. Meanwhile, the counter of the timer will be loaded automatically with the next value, that should be readily available in the 8253 chip. So the software should think one step ahead. This timer constitutes the reference of absolute time.

As soon as an interval between two stations (DTA) has passed, we should generate 8 bursts. By that time, the hardware must have been set up for these eight bursts (signal strength, ECD, etc.). From burst to burst, only the phase code will change, according to the following scheme:

	Master	Slave
GRI A:	+ + - - + - + -	+ + + + + - - +
GRI B:	+ - - + + + + +	+ - + - + + - -

The generation of the correct phase codes will take place by means of a phase code table (PHCTAB), in which all phase codes are stored. By rotating a code bit-by-bit we find the correct + or - code (1 or 0). After all eight bursts, the code will be adjusted for the next GRI by inverting the sign of all *even* numbered bursts.

Setting up the hardware is critical: it is allowed to set various parts of the hardware only if there is no signal to be generated. The main timer must be controlled correctly meanwhile. To explain all actions that should be taken for one station, consider the following illustration:



Two variables keep track of the "state" of the simulator: INDEX stores the number of the station that is processed at the moment (master = 0, slave 1 = 1 etc.). This variable is used, among others, for indexing various tables. BRSCNT is a burst counter for the eight bursts of a station. It counts down from 7 to 0.

After every interrupt the sign switch will have to be prepared according to the phase code of the following burst. After the 7th burst of a station, the main timer must be loaded with the DTA that comes after the last burst. At the 8th burst this value will be loaded in the timer automatically.

The DTA's have to be divided into 10 μ s units for the main timer, 200 ns units for the coarse delay line and 25 ns units for the fine delay line. The main timer keeps track of absolute time. The DTA will be split into parts using the following algorithm:

```

timervalue = DTA / 10  $\mu$ s
remainder = old remainder + (DTA mod 10  $\mu$ s)
if remainder > 10  $\mu$ s then ---> remainder = remainder - 10  $\mu$ s
                                timer = timer + 10  $\mu$ s
coarse delay = remainder / 200 ns
fine delay = remainder mod 200 ns
  
```

The relative moves in time will be kept in the variable REMAINDER. All previous relative shifts are added together; as soon as the 10 μ s threshold will be exceeded, the timer value and REMAINDER will be corrected. The remainder will be split up into 200 ns parts and 25 ns parts. for the delay lines.

When all calculations have been done, the program will check whether there is a request for a relative move of the stations (CHK_MOV). If the semaphore of the transmitter in question is set (MOV_FLG), the transmitter will be moved for the next GRI. The DTA's on both sides will be adjusted for this.

When the next interrupt request arrives from the timer, all processing for the current transmitter has been done and we can start preparing the next transmitter. Before the hardware will be switched, we will wait for 1 ms to let the last burst fade out.

BRSCNT will be set to 7 again, and the timer can be programmed for a 1 ms interval for the next series of bursts. Next, the phase code of the previous station will be set for the next GRI. The new phase code will be stored in PHCODE, which is used for rotating the sign bits.

The ECD's and signal levels of ground- and skywaves can then be set, and we can set the delay lines according to the value of REMAINDER. All hardware now is ready for the next station. All we have to do is wait for the next interrupt

One important remark: before programming the delay lines, we must switch off the carriers to prevent phase modulation of the low-pass filters. Phase modulation of the low-pass filters could cause high voltage peaks, that can leak through the LOGDAC and "infect" the output signal. After that we have to wait 500 μ s before the transients have faded out. We can switch on the carriers again after this period.

3.3.6 Test module (SIMTEST.MAC)

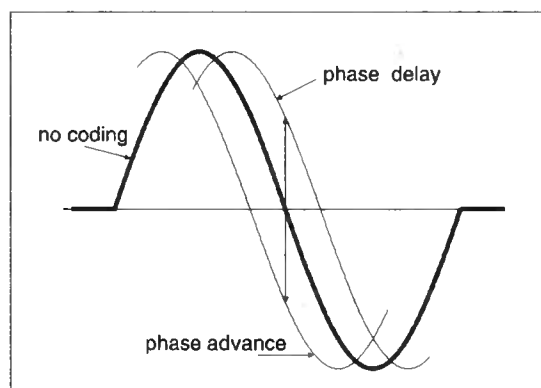
At start up, SIM.MAC will check whether it should run a test program. The user could have set dipswitches to order the simulator to do so. The test programs will generate various signals to check whether the simulator is operating correctly. Some test programs are needed to adjust various resistors and capacitors in the simulator itself. These test programs are not very interesting any more, so I won't discuss them in this report.

4. Modification of a Loran-C simulator for Data Transmission

4.1 Introduction

You now know what Eurofix is, and that we need a data-link for transmitting the DGPS corrections to the (mobile) user. In Eurofix, data transmission is achieved by shifting Loran-C bursts with respect to each other within a group of eight bursts. As said in Chapter 2, research is still going on and nothing is decided yet on the final form of Eurofix. However, the need was felt to have a Loran-C simulator at our disposal that could generate Eurofix signals in a premature form. This, together with a modified receiver that could decode our premature modulation scheme, would offer us the possibility to show to sceptics that Eurofix can really work. Even if premature, speedy results are important in order to convince the authorities that Eurofix is interesting to implement in the planned new transmitters in Europe. Besides that, modifying a Loran-C receiver for Eurofix in whatever form without the use of a suitable simulator is practically impossible. It would also enable us to show that conventional receivers don't suffer too much from Eurofix.

In the "premature" modulation scheme, the last 6 bursts of a group of eight as shifted as depicted below:



Binary data
bit value

Modulation pattern

1	0	0	+	-	+	-	+	-
0	0	0	-	+	-	+	-	+

Modulation pattern for code rate of 6

0 = no time shift

+ = positive code bit = 1 μ s phase delay

- = negative code bit = 1 μ s phase advance

In this way it is possible to transmit 1 bit per station per GRI.

4.2 Design considerations for the Eurofix simulator modifications

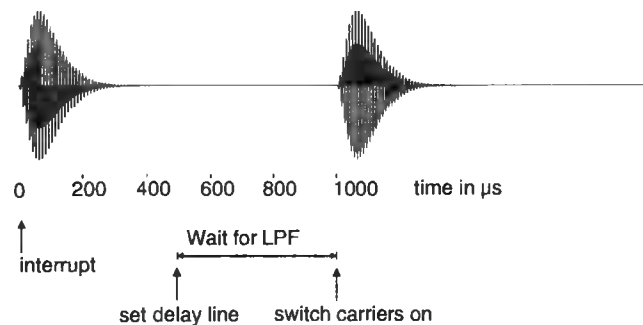
The shifts of 1 μ s are taken out of the blue. No research has been done on this value. It is clear that a larger shift can increase the Signal to Noise ratio of the data signal, but a larger shift will also mean a larger decrease in Signal-to-Noise ratio of the Loran-C signal for navigation purposes. The US Navy once had their Clarinet Pilgrim system, also based on phase modulation of Loran-C signals. The datalink they set up in that way was used to guide submarines. The US Navy also used a modulation index of 1 μ s, which turned out to be satisfactory. So this seemed a nice value to start with. But complete other modulation schemes may be feasible as well. With ternary coding (a + shift, a - shift or no shift at all) we might achieve a bit-rate that is factor 1,5 larger, at the cost of an increase in Bit Error Probability. So, it is not yet clear what the optimum will be.

Because the Eurofix system, outlined in the previous section, is only a premature proposal, care should be taken to modify the simulator in such a way that it allows other proposals to be implemented easily.

Grosso modo the Loran-C simulator is a signal generator that can generate Loran-C bursts, at times dictated by a microprocessor controlled module. Because in Eurofix only the time of transmission of Loran-C bursts will differ for normal Loran-C, we could expect that the modifications could be made solely in the control software of the simulator. This will mean that the simulator is still suitable for "normal" Loran-C simulations and that it will be a relatively easy way to implement other modulation schemes. So, it would be important to try to keep modifications restricted to the software of the simulator as much as possible. Furthermore, it was not the intention to modify the simulator in such a way that it couldn't be used for "normal" operations any more.

Assuming that the modifications could be limited to software, it would then be a nice idea to keep the software as far as possible compatible with older PC control software.

Intuitively, the biggest problem will be that the Time Of Transmission of burst 2..8 of a station can vary. Up until now, the timer could be set to 1 ms, and one could sit down and wait. The calculation of Time Of Transmission had only to be made for the DTA's, once per station. For Eurofix, such a calculation should be made for *every* burst. A quick glance at the original module "SIMBRS.MAC", in particular at the procedure NEXT_DTA, learns that at least a 32 bit integer division is needed. Divisions are known to be time consuming, so an investigation of the subroutine "Div32" tells us that (worst case) 350 μ s are needed. Another problem is the not-allowed phase modulation of the Low-Pass filter: before we can program the delay lines, we must switch off the carrier. After programming the delay-lines, we must wait 500 μ s to let transients fade out, before we may switch on the carrier again. And, of course, before we are allowed to program the delay lines, the current burst must have faded out. Because we only have 1000 μ s and we have a lot to do, there is reason to worry! This problem may become clear from the following figure:



At time $t = 0$, the main timer will generate an interrupt, at which the program will start an interrupt service routine. In this service routine, the delay lines for the *next* burst will have to be set, and the timer value for the burst *after that* will have to be calculated and set. This interrupt signal will also be noticed by the delay line and the "synchroniser". At time $t = 0 + t_{\text{delay}}$, the actual Loran-C burst will start. Before we are allowed to reprogram the delay line, the Loran burst must have faded out, so we must keep our hands from the delay line until $t \approx 300 \mu\text{s}$. Starting from $t = 0$, we can calculate the timer value for the burst after the next, which will take about $350 \mu\text{s}$ (the infamous 32 bit division). After that, the current Loran burst will have faded out, and we can reprogram the delay line. But we must reprogram the delay line before $t = 500 \mu\text{s}$, in order to let the Low-pass filter recover. During this $500 \mu\text{s}$, we cannot do anything else. So, we don't have much time left to take care of other engagements (PC communication, setting signal levels, ECD's, skywave parameters, dynamic simulation etc.).

Another major modification is to take care of the data flow from PC to simulator. Problems will arise for sure if you try to regulate this flow via the command buffer of the simulator. Commands in the command buffer are carried out as soon as the microprocessor has time to do so. No guarantees can be given that a certain command is carried out at a certain time. For dynamic simulation of "normal" Loran receivers, this doesn't pose a problem. Calculating a position fix in a receiver is a "slow" procedure. A receiver cannot distinguish between a change in (for instance) skywave level in GRI n or GRI $n+1$.

4.3 Two major modifications in the Loran-C simulator

Making the Dynamic Dual-Chain Loran-C Simulator suitable for Eurofix, requires some modifications that can be divided into two major classes. First of all, the simulator must be able to receive the data it should transmit from the PC. The communication protocol has to be extended to allow for this. This data has to be put in some buffer.

Secondly, the simulator must be able to change the Time Of Transmission of *any* burst, according to the data it received from the Personal Computer.

4.3.1 Buffering of data bits

I will start with the simplest problem outlined in paragraph 4.2: the handling of the data flow from PC to simulator, especially the DGPS data. When the DGPS data from the PC would be put into the regular command buffer, you lose flexibility. For instance, it is not possible anymore to load a simulation scenario (a route the simulator should simulate, incl. varying

ECD's, SNR's and sky waves) into the command buffer and to start the simulator. All problems will be circumvented by intercepting the DGPS data from the RS-232 flow and placing this data into separate buffers, which is exactly what has happened.

In the data-area of the simulator software, 7 buffers of 16 bytes each have been added to store DGPS data, for each station one buffer. The module "SIMIO.MAC" is somewhat extended. When a character from the PC is received, the interrupt service routine checks a variable to see whether there are (DGPS) data bytes expected. If not, it will continue its "original" tasks.

If the PC wants to send data bytes, it first sends a command to tell the simulator that data bytes will follow. The simulator now sets a flag to know that data bytes will follow. After that, the PC will send a station identification number, to identify the station for which the data is destined. In order to save overhead trouble (data-command, station ID), the PC will send 4 bytes (of 7 bits each), which will be stored into the appropriate buffer.

Because both chains listen to the same RS-232 line coming from the PC, chain A might recognise its "SELECT" command in the DGPS data for some station in chain B. This problem is circumvented by letting the not-selected chain also count the number of data bytes its colleague is expecting. The one chain now knows that a certain character is a data byte for the other chain and not a "SELECT" command for itself.

4.3.2 Setting the timer and delay lines

As sketched in paragraph 4.2, the simulator will be very busy carrying out its tasks, maybe even too busy. Not only will the simulator have to do its regular tasks, but it will also have to carry out its additional tasks, like processing an increases flow of commands and data from the PC. Not only for itself, but also for its colleague as all bytes destined for its colleague also have to be read to look for a "SELECT" command too.

We could cut a large part of the tasks of the simulator by getting rid of the division it needs to make for every burst. The timing of the transmission of burst is based on a timer and a delay line. The timer can create delay in units of 10 μ s, the delay line can delay signals 0 .. 9.975 ns in steps of 25 ns. The total delay time has to be divided by 400, in order to split it up into a part in units of 10 μ s and a part in units of 25 ns. But if a DTA is already divided in parts, and if we calculate time delays in steps, we don't need this 32 bit integer division any more! Calculations can be limited to simple additions and subtractions.

It took me some time to fully understand the timing structure of the simulator, so I don't expect you to understand at this point. I will try to clarify it with an example:

Suppose we want to transmit the following TD's (static, and without Eurofix modulation) on a chain with a GRI of 7970 (repetition time is 79.700 μ s).

TD master - slave 1: 16975.225 μ s	DTA master - slave 1 = 9975.225 μ s
TD master - slave 2: 26908.175 μ s	DTA slave1 - slave 2 = 2932.950 μ s

The simulation will start with the second burst of the master (it will become clear why). Therefore, the *timer* is set to 100, i.e. 100 units of 10 μ s. The *delay line* is set to 0. Before the start of the last burst, the timer is set to the DTA master-slave 1. When the timer is ready counting the interval between burst 7 and burst 8 of the master, it starts counting down the interval DTA. The *timer* will be loaded with the number 997. The *delay line*

will then be set to the number 209 (in 25 ns units). After that, the interval between two consecutive bursts of slave 1 have to be timed: the *timer* will be set to 100 again, the *delay line* will keep its value, until the next DTA has to be timed.

The values for *timer* and the *delay line* for DTA 2 are calculated as follows (following the algorithm in paragraph 3.3.5):

$DTA\ 2 = 293 * 10\ \mu s + 118 * 25\ ns = 2932.950\ \mu s$ (division needed to split 2932.950 into parts!)

$delay\ line = delay\ line + 118;$
 $timer = 293$

If the result of the first addition would have been bigger than the range of the delay line, the timer value would have been 294.

For static simulation, the DTA's remain constant, for dynamic simulation, we only slightly modify them once in a while. The problem of time-consuming divisions can be solved by storing and using the DTA's in split form! So, if the PC could split the DTA's for the simulator, no division are necessary. But, as stated before, if the simulator software is to be kept compatible with PC software, this is no option. A solution is then to split the DTA's upon reception and store them in split form. The divisions have then moved to the initialisation part of the simulator, so they aren't necessary anymore during the course of the simulation.

At this point I hope it is clear to you that the *absolute* time within a GRI is kept in two variables: *timer value* and *delay line value*. You will find these variables back in the source code of the simulator *quotient* and *remain* respectively.

Modulation of Eurofix data onto the bursts is an easy task now: if a burst has to be transmitted somewhat earlier, the calculated delay time (without Eurofix) has to be decreased a little (temporarily). The source code of the modified simulator software can be found in Appendix C. If you take a look at the procedure SET_TIMER in the module "SIMBRS.MAC", you will find that the absolute time within a GRI is kept in the variables "quotient" and "remain". These two variables together specify the Time Of Transmission of the next (unmodulated) burst. Both will be adjusted temporarily to get the actual time of transmission which differs because of Eurofix.

The waiting period of 500 μs , that would be necessary to prevent phase modulation, appeared to be a rather conservative value. Measurements with storage oscilloscope showed that nothing could be found if the period was decreased to 200 μs

4.4 Changes in the original software

The structure of the modifications necessary for Eurofix has been outlined in the previous section. In appendix A you will find a detailed description of the additional commands a Personal Computer may issue for Eurofix. There are commands for transmitting data from PC to simulator, and commands to set various Eurofix parameters as modulation pattern and modulation index. I will now discuss the changes made in every module.

4.4.1 Communication module (SIMIO.MAC)

The simulator has 7 extra buffers (one for each station), in which data coming from the PC will be stored. Because all buffers are read out in the same pace, only 1 read pointer will be necessary for all buffers. However, all buffers will need a separate write pointer, because the PC is allowed to fill the buffers in any order it likes. The 7 write pointers and 1 read pointer will be initialised in routine INITRS.

The interrupt handler routine for a RS-232 interrupt will now intercept data bytes that are to be modulated onto the Loran-C bursts. The intercepted bytes will be handed over to the procedure "data bytes". Even if the chain in question is not selected. Besides that, the interrupt service routine hasn't changed very much.

As said above, data bytes will be handed over to the procedure "data bytes". The PC will send a "data command" to tell the simulator that data bytes will follow. "Data bytes" will then set a variable called "dataexp" to 5, so that the simulator knows that the coming 5 bytes are data bytes and not command bytes. This is to circumvent the problem that the data bytes might contain some character that is recognised as a command. The data bytes will be written to the appropriate buffer.

Even if the simulator is not selected, it start counting data bytes after it sees a data command on the RS-232 line. When scanning the line, it might find its "SELECT" command in the data bytes for the other chain. By counting the data bytes for the other chain, it can distinguish between data bytes for the other chain and the "SELECT" code for itself.

The routine "SEND" now checks whether the simulator is selected. Formerly, the simulator could only send data on request of the PC, so it must had been selected. In the Eurofix mode of operation, the simulator will send a synchronisation character after each 112 GRI's, which is only allowed if it is selected to prevent "data collisions" on the RS-232 line.

4.4.2 Main module (SIM.MAC)

The command set of the simulator is extended somewhat to make Eurofix possible. All commands that the simulator will understand, can be found in Appendix A. The additional Eurofix "ESC"-commands are decoded and carried out in this module. The changes in this module speak for themselves.

4.4.3 Burst processing module (SIMBRS.MAC)

The Times of Transmission of bursts will have to be modified, according to the data the PC want the simulator to transmit. This data is put into buffers by procedures in the module SIMIO.MAC. The actual modulation takes place in this module.

Let me start with explaining the routine SET_TIMER. In this routine, the settings for the timer and the delay lines without Eurofix modulation are calculated. The settings for timer and delay lines are added to "quotient" and "remain", so these variables together contain the *absolute* time reference within a GRI. These variables are compensated for Eurofix. These compensated variables are called "tempquo" and "temprem". Timer and delay lines will be eventually be set with these values. After SET_TIMER has calculated those values, the timer will be set right away (for the burst after the next).

The conductor of all timing procedures is "BRSINT". This interrupt service routine will get called after the timer reaches zero. This interrupt signal is also a signal for the burst generator to start generator a burst (after a delay). So, the burst will begin at the same time this routine is called.

This routine calls SET_TIMER to set the timer for *the burst after the next*. Furthermore, the delay lines will be set for the *next* burst. But this is only allowed if the current burst (that is just being generated) has finished.

BRSINT also keep tracks of the number of bursts per station. The variable "brsct" is used for this purpose. If all eight bursts of a station have been transmitted, the timer must have been set to wait a period, the DTA. So, BRSINT also tells SET_TIMER to set the timer to time inter-burst periods or inter-station intervals.

Because BRSINT is not allowed to set the delay lines before the current burst is finished, it can do its calculation work in the mean time. For that reason SET_TIMER is already called to calculate a new "tempquo" and "temprem". This will take a while. After that, the delay lines can be set to the value that was saved in "temprem2".

At the start of BRSINT, the variable "timeleft" is loaded with a value of 56. This value would cause the waiting loop in SET_DEL to wait about 280 μ s. By decreasing this value now and then after some tasks, SET_DEL knows how much work has been done before it was called and knows how much time to wait in addition before 300 μ s have passed since the begin of the burst. The amount of time BRSINT spends before it calls SET_DEL can be determined in this way and SET_DEL doesn't have to wait too long, nor will it wait too short.

Speaking of SET_DEL, SET_DEL sets the delay lines to the value in variable "temprem2". But it doesn't do that until at least 300 μ s have passed since the beginning of the current burst. SET_DEL will also set the skywave timer to the appropriate value. The actual setting of the delay lines is postponed to the end of this procedure, in order to limit the time to wait to a minimum. The device is to spend as little time as possible in the waiting loop.

When you compare the original "SIMBRS.MAC", you will find a new procedure: PREP_NEXT. In this routine, the Eurofix modulation parameters are determined. PREP_NEXT will get the next bit to be modulated onto the bursts of the next station from the table, and will fill in the appropriate modulation pattern for this bit into the variables that store that pattern. For each station there is an entry in "patt_tab" and "shft_tab". These variables can be compared to the "phcode" variable. At the beginning of a new cycle of eight bursts, the pattern and shift pattern are filled in, and they are rotated during the cycle. Each bit in the pattern table corresponds to a burst in the group of eight. A zero-bit in this pattern tells SET_TIMER not to modulate the burst, a one-bit tells SET_TIMER to shift the burst, in the direction that is determined by the corresponding bit in the shift pattern. A glance at the source code will make it clear to you.

4.5 Performance analysis of the modified Loran-C simulator

In this paragraph I will try to convince you that the simulator can work, that it is able to carry out all the tasks it has to do. Therefore I have counted the number of clock-states various routines will take when they follow various paths. These values are shown in the following table

This calculation holds for 1 station. For 3 stations, the number will have to be tripled:

$$= 702 \text{ states}$$

This total of 9672 states takes 19.3 ms in a GRI of about 80 ms. So, there is time enough left to take care of other tasks as relative moves, dynamically changing signal levels, ECD's etc.

4.5 Modifications in a nutshell

SIM.MAC

The module "SIM.MAC" is extended with commands to set the modulation pattern for zero- and one bits, to set the magnitude of a phase advance or phase delay and to set the mode of operations for all stations. These subroutines are:

Command:	Name:	Subroutine
ESC L	Set zero pattern	set_0patt
ESC M	Set zero shifts	set_0shft
ESC N	Set one pattern	set_1patt
ESC O	Set one shifts	set_1shft
ESC P	Set phase advance	set_advnc
ESC Q	Set phase delay	set_delay
ESC R	Set Eurofix mode	set_eurof

These subroutines speak to themselves. A detailed description can be found in Appendix A.

SIMIO.MAC

The module "SIMIO.MAC" has some facilities to intercept DGPS-data from the PC from the regular flow of commands from the PC. This DGPS data is placed in separate buffers, each station has its own buffer. Data should be sent in strings of 28 bits. All of the data-interception is done in the subroutine "data bytes". The subroutine "data bytes" also tracks data bytes for the other chain, in order to distinguish between data bytes for its colleague and "SELECT" commands for itself. To count how many data bytes can be expected, "data bytes" uses the variable "dataexp".

SIMBRS.MAC

Most of the changes for Eurofix were necessary for this module. People who know the original "SIMBRS.MAC" will look for "next_dta" in vain. The functions of this subroutine can be found back in the more general "set_timer", in which all timer programming is done. The former "next_stat" has surrendered the privilege of setting the timer. The subroutine PREP_NEXT" is new. This subroutine calculates the index of the following station in chain, reads data bits from the buffer and sets ready the modulation pattern, that is to be used for the coming GRI. Control of the carrier, phase code and ECD bits is left over to CARRIERS_ON.

A. Operation manual Dynamic Dual-Chain Loran-C Simulator

The operation of the Dynamic Dual-Chain Loran-C Simulator is completely controlled by software commands that can be fed into the simulator via an RS-232 interface: the simulator doesn't have any keys, except for the On/Off switch. In this section you will find a list of commands and their description.

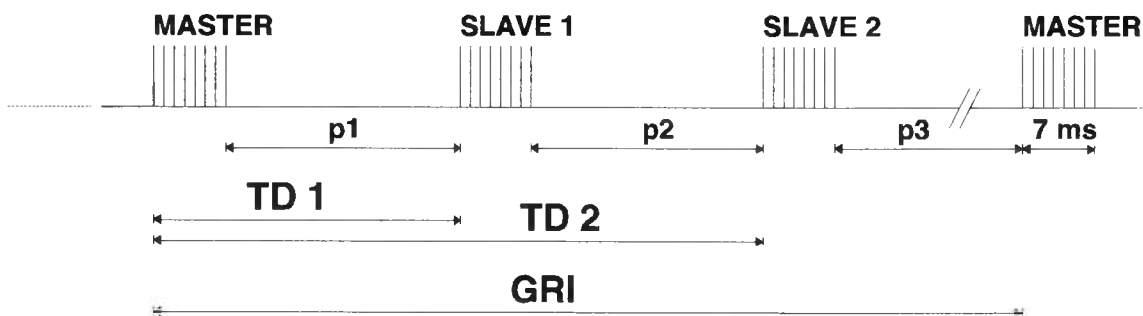
The commands you can give the simulator can be divided into two categories: direct commands and buffered commands. Buffered commands are placed in a buffer. The simulator will process them when it is time to do so. The direct commands are carried out right away, before any other command in the buffer.

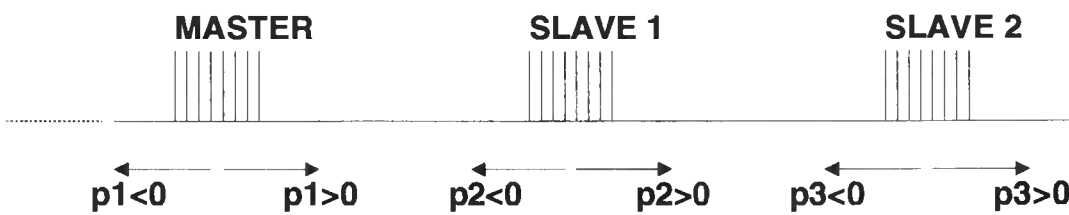
Direct commands

command	format
SELECT	1E (hex) for chain A 1F (hex) for chain B
<p>You can use this command to select chain A or chain B of the simulator. Because the two separate Loran-C signal generators in the simulator share one communicationline, you need to tell for which chain the commands are. A chain is selected until you send a select-command for the other chain. The signal generator that is not selected continues to work of course.</p>	
command	format
STATUS	1C (hex)
<p>This command causes the selected chain to return the status of its commandbuffer in 2 bytes (first the Least Significant Byte, then the Most Significant Byte). The number that the selected simulator will return is the number of bytes present in its commandbuffer.</p>	
command	format
START	19 (hex)
<p>After reception of this command, the selected simulator will start generation of Loran-C bursts. This offers the possibility to load the commandbuffer of the simulator with commands before the signal generation will start.</p>	

Buffered commands

command	format
SET NUMBER OF TRANSMITTERS	ESC A <p>
<p>With the sequence "ESC A <p>" you can order the simulator to simulate a chain of <i>p</i> transmitters. The maximum is 7 (1 master, 6 slaves). You must send this command before any other command, so that the simulator knows how many parameters to expect for other commands. During the course of the simulation you cannot change the number of transmitters.</p>	

command	format
SET INITIAL TD's	ESC B <p1> <p2> <p3>
<p>The initial TD's will have to be specified in DTA format, i.e time between last burst of some transmitter and first burst of the next transmitter in the chain. How DTA's, TD's and GRI depend upon each other can be seen from the figure below, in which the timing of a chain with 1 master and 2 slaves is shown. When all parameters <i>p1</i> .. <i>p3</i> are chosen, the simulator will know the GRI to operate on.</p>  <p>All parameters are specified with a resolution of 25 ns. The maximum for each parameter is about 4.000.000 (≈ 100 ms). You can set the simulator to operate on any GRI, existent or non-existent in real life.</p>	

command	format
SET RELATIVE MOVES	GS <p1> <p2> <p3>
<p>Dynamic simulation is possible with the use of this command. By specifying the initial TD's, all transmitters find their place in the time-frame, and the major timing procedures (GRI) of the chain will depend on these initial TD's. With this command we can move the positions of the transmitters within the time-frame. A positive parameter means a move to the right (later in time), a negative parameter means a move to the left (earlier in time). A zero parameter keeps the transmitter one the same place.</p>  <p>The parameters are specified in units of 25 ns. The range of each of the parameters is -128 .. 127, so you can have a maximal shift of $127 * 25 \text{ ns} \approx 3 \mu\text{s}$. These commands are internally synchronised to the transmitted GRI, so that the relation with absolute time is established.</p>	

command	format
SET SKYWAVE DELAYS	ESC D <p1> <p2> <p3>
<p>When skywaves have to be generated, you can set the skywave delay for each of the stations with this command.</p> <p><p1> = skywave delay master <p2> = skywave delay slave-1 <p3> = skywave delay slave-2</p> <p>These parameters have a resolution of 25 ns and a range of 0...700 μs</p>	

command	format
SET GROUNDWAVE LEVELS	ESC E <p1> <p2> <p3>
<p>With this command you can set the groundwave levels of each of the stations.</p> <p><p1> = groundwave level master <p2> = groundwave slave-1 <p3> = groundwave level slave-2</p> <p>The parameters specify the attenuation with respect to the maximum output level (= 0 dB). The resolution of this attenuation is 0.375 dB. The range is 0 (0 dB attenuation) to 240 (90 dB attenuation).</p>	

command	format
SET SKYWAVE LEVELS	ESC F <p1> <p2> <p3>
<p>You can set the skywave levels of each station with this command.</p> <p><p1> = skywave level master <p2> = skyave slave-1 <p3> = skywave level slave-2</p> <p>The parameters specify the attenuation with respect to the maximum output level (= 0 dB). The resolution of this attenuation is 0.375 dB. The range is 0 (0 dB attenuation) to 240 (90 dB attenuation).</p>	

command	format
SELECT POLARITY	ESC I <p>
<p>The polarity of the output of the selected signalgenerator can be set using this command. The cycle-identification process of some receivers might have difficulties if the polarity of the signal is reversed. Using this command you can circumvent that problem.</p> <p><p> = 0 : positive polarity (default) <p> = 1 : negative polarity</p>	

command	format		
SET GROUNDWAVE ECD	ESC G <p1> <p2> <p3>		
You will need this command to set the Envelope-to Carrier-Delay of the groundwave of each of the stations. The parameters are specified as follows:			
ECD (in μs)	parametervalue	ECD (in μs)	parametervalue:
	0	0.5	11
-5.0	1	1.0	12
-4.5	2	1.5	13
-4.0	3	2.0	14
-3.5	4	2.5	15
-3.0	5	3.0	16
-2.5	6	3.5	17
-2.0	7	4.0	18
-1.5	8	4.5	19
-1.0	9	5.0	20
-0.5	10		
0			

command	format
SET SKYWAVE ECD	ESC H <p1> <p2> <p3>
See "SET GROUNDWAVE ECD"	

command	format
SET NOISE LEVEL	ESC J <p>
<p>Chain A of the simulator is equipped with a white-noise generator. The outputlevel of this generator is adjustable with this command. The parameter <p> specifies the attenuation with respect to the maximum outputlevel of the noisegenerator (= 0dB) in 0.375 dB units. The range extents from 0 (-26 dB noiselevel) to 236 (-114.5 dB noise level).</p>	

command	format
RESET SIMULATOR	ESC K
<p>Issuing this command will reset the selected simulatorpart. Signalgeneration is stopped and all parameters will take their default values again. It is then possible (for instance) to select a new GRI.</p>	

Eurofix commands

command	format
SET EUROFIX ZERO PATTERN	ESC L <p>
<p>When the simulator is going to work in the Eurofix mode, you can specify which of the bursts of the group of eight of each station will have to be modulated when a zero-databit has to be transmitted. You will have to set the Most Significant bit of the eight-bit parameter <p> to 1 if you want the first burst of the group of eight to be modulated, and so on for the rest of the burst. Normally you will want to leave the first two bursts unmodulated (because they are important for blinking-purposes and you don't want the receiver to miss them). So the parameter will normally be 00111111 (binary).</p>	

command	format
SET EUROFIX ONE PATTERN	ESC N <p>
<p>You can set the modulation pattern for a one-databit seperately, in the same way as is described for "SET EUROFIX ZERO PATTERN". To be clear: by setting the pattern you only specify which of the bursts will have to be modulated. You can tell the simulator <i>how</i> the bursts will have to be modulated using the following commands:</p>	

command	format
SET EUROFIX ZERO SHIFT	ESC M <p>
<p>Assuming that you already have decided which bursts of the group of eight you want to shift to transmit a zero-databit, you can specify with this command in which <i>direction</i> each of those bursts should shift. Again, the most significant bit of parameter <p> belongs to the first burst. If this MSB is a one, the burst will be advanced, for a zero the burst will be delayed. The value of a bit of <p>, belonging to a burst that is not going to be modulated according to the pattern, doesn't care. To be completely clear: "SET EUROFIX ZERO PATTERN" and "SET EUROFIX ZERO SHIFT" <i>together</i> define the modulation pattern of advanced and delayed bursts for a ZERO-databit, i.e. which burst will shift which way. How much the bursts will shift, can be set with "SET EUROFIX PHASE ADVANCE" and "SET EUROFIX PHASE DELAY".</p>	

command	format
SET EUROFIX ONE SHIFT	ESC O <p>
<p>Assuming that you already have decided which bursts of the group of eight you want to shift to transmit a one-databit, you can specify with this command in which <i>direction</i> each of those bursts should shift. Again, the most significant bit of parameter <p> belongs to the first burst. If this MSB is a one, the burst will be advanced, for a zero the burst will be delayed. The value of a bit of <p>, belonging to a burst that is not going to be modulated according to the pattern, doesn't care. To be completely clear: "SET EUROFIX ONE PATTERN" and "SET EUROFIX ONE SHIFT" <i>together</i> define the modulation pattern of advanced and delayed bursts for a ONE-databit, i.e. which burst will shift what way. How much the bursts will shift, can be set with "SET EUROFIX PHASE ADVANCE" and "SET EUROFIX PHASE DELAY".</p>	

command	format
SET EUROFIX PHASE ADVANCE	ESC P <p>
<p>Databits are modulated onto the Loran-C signal via a pattern of burst delays and advances. These patterns can be set using the foregoing four commands, for zero- and one-databits independently. If a burst has to be advanced according to the modulation pattern, you can use this command to tell the simulator <i>how much</i> it has to advance that burst. The parameter <p> has a resolution of 25 ns. So, for a typical phase advance of 1 μs, you would set <p> to 40.</p>	

command	format
SET EUROFIX PHASE DELAY	ESC Q <p>
<p>By now you will suspect what this command is going to be used for: setting the time that a burst will come later when the modulation pattern specifies so. Again: <p> has a resolution of 25 ns.</p>	

command	format
SET EUROFIX MODE	ESC T <p1> <p2> <p3>
<p>This command allows you to specify for any station in the chain whether it should operate in Eurofix mode (modulating data bits onto the bursts) or not. The parameter <p> corresponding to the station should be 0 if the station has to operate without Eurofix, FF (hex) if the station should operate in Eurofix mode.</p>	

command	format
SEND DATA	1A <n> <xx> <xx> <xx><xx>
<p>Data that the simulator will have to modulate onto the Loran-C signal in Eurofix mode can be fed into the simulator by using this command. After the command "1A", you must tell the simulator for which station the following 4 bytes of data (7 bits per byte, so 28 bits) are intended. This allows you to update the databuffer of all stations in any order. As long as you make sure to send 28 bits at a time! The simulator will modulate the bits in "raw" form, it doesn't need knowledge on the format, framing etc.</p>	

command	format
SYNCHRONISATION	SYN
<p>The simulator is not completely mute: it can send information by the RS-232 link as well. Every 112 GRI's, the simulator will send the ASCII "SYN" (16 hex) character. The data that the simulator has to modulate onto the bursts will be buffered by the simulator to avoid race conditions. But the Personal Computer that is controlling the simulator must have some sense of time: it should avoid that the databuffer runs out of data, and it should also avoid a buffer overflow. Because the PC knows on what speed bits are transmitted, and because it has an internal clock, it can take care of filling the buffer in time. However, on the long term the clocks in the simulator and in the Personal Computer will drift away from each other. The Personal Computer can resynchronise its clock after reception of this character.</p>	

B. Source code listings of original software

MACRO-80 3.44 09-Dec-81

PAGE 1

```

*****
;*
;*   Loran-C Dynamic Simulator Main Program
;*
;*   by R.Kellenbach
;*
;*   Copyright (C) '86 by the University of Delft
;*
;*   Created   : nov  1. 1986
;*   Last update : feb  1. 1988
;*
*****

0018          ESC      equ    1bh          ;escape code
001D          GS       equ    1dh          ;rel. move command code

0080          intc11    equ    80h          ;interrupt clear flip-flop's
0081          intc12    equ    81h

0091          uartc     equ    91h          ;8251 status register
00A0          swport    equ    0a0h        ;dip switch input port

0086          noisep    equ    86h          ;noise control port

          .z80
0000*         cseg

          .request simio,simbrs,simtest

0000*  31 0080*         ld      sp,stack      ;init stack

0003*  CD 0000*         call   initrs##       ;init RS-232 interface

0006*         boot:                ;restart address

0006*  CD 0000*         call   initbrs##      ;init burst processing

0009*  3E 01           ld      a,1           ;start noise generator
000B*  D3 86           out     (noisep),a
000D*  3E 00           ld      a,0
000F*  D3 86           out     (noisep),a

0011*  DB A0           in      a,(swport)    ;get switch status
0013*  2F              cpl
0014*  E6 0F           and     0fh          ;switch 1..4
0016*  C2 0000*        jp      nz,test##     ;hardware testing ?

0019*  ED 56           im      1           ;interrupt mode 1
001B*  FB              ei                 ;enable interrupts

001C*  C3 0053*        jp      get_cmd      ;skip interrupt vector

*****
;*
;*   Subroutine :   Interrupt
;*
;*   Function :    Uart & Timer interrupt
;*
```

MACRO-B0 3.44 09-Dec-81 PAGE 1-1

```

; *                               *
; *   Inputs :   interrupt flags in uart status   *
; *                               *
; *   register   *
; *   Outputs :   none                           *
; *                               *
; *                               *
; *****

; org 38h ;mode 1 interrupt vector

0038' F5      push  af      ;save registers
0039' C5      push  bc
003A' D5      push  de
003B' E5      push  hl

003C' 21 0040' ld  hl,exit_int ;push return address on stack
003F' E5      push  hl

0040' DB 91      in  a,(uartc) ;get interrupt flag bits

0042' CB 7F      bit  7,a      ;burst timer interrupt ?
0044' CA 0000*   jp   z,brs_int##

0047' CB 4F      bit  1,a      ;uart RXRDY interrupt ?
0049' C2 0000*   jp   nz,rx_int##

004C' C9      ret      ;other interrupts not allowed

004D'          exit_int:      ;exit interrupt routine
004D' E1      pop   hl      ;restore registers
004E' D1      pop   de
004F' C1      pop   bc
0050' F1      pop   af
0051' FB      ei          ;enable interrupts again
0052' C9      ret

; *****
; *                               *
; *   Subroutine :   Get_cmd   *
; *   Function :   Main program command processing *
; *   Inputs :   commands in input data buffer   *
; *   Outputs :   burst procesing data tables   *
; *               updated   *
; *                               *
; *                               *
; *****

0053'          get_cmd:
0053' 21 0053'   ld  hl,get_cmd ;push return address on stack
0056' E5      push  hl

0057' CD 0000*   call get_byt## ;get command character

005A' FE 1D      cp   GS      ;rel. move command ?
005C' CA 00FD'   jp   z,rel_mov

005F' FE 1B      cp   ESC     ;ESCAPE command ?
0061' CA 0065'   jp   z,esc_cmd

```


MACRO-80 3.44 09-Dec-81 PAGE 1-2

```

0064' C9                                ret                                ;invalid command, ignore

0065'                                esc_cmd:                                ;escape command processing
0065' CD 0000*                          call    get_byt##                    ;get command character
0068' FE 41                             cp      'A'                        ;command A-Z ?
006A' D8                                ret      c
006B' FE 5B                             cp      'Z'+1
006D' D0                                ret      nc

006E' D6 41                             sub      'A'                        ;perform table look-up
0070' 87                                add      a,a
0071' 5F                                ld      e,a
0072' 16 00                             ld      d,0
0074' 21 007D'                          ld      hl,cmd_tab
0077' 19                                add      hl,de
0078' 7E                                ld      a,(hl)
0079' 23                                inc      hl
007A' 66                                ld      h,(hl)
007B' 6F                                ld      l,a
007C' E9                                jp      (hl)                        ;jump to selected function

007D'                                cmd_tab:                                ;escape command address table
007D' 0120'                             dw      no_stat                    ;ESC A = set # stations
007F' 0128'                             dw      set_dta                    ;ESC B = set initial dta's
0081' 00B1'                             dw      invalid                    ;ESC C = not used
0083' 0145'                             dw      set_sdel                    ;ESC D = set skywave delays
0085' 0150'                             dw      set_glev                    ;ESC E = set gndwave att. level
0087' 0170'                             dw      set_slev                    ;ESC F = set skywave att. level
0089' 01B3'                             dw      set_gecd                    ;ESC G = set gndwave ECD
008B' 0196'                             dw      set_secd                    ;ESC H = set skywave ECD
008D' 01A9'                             dw      set_sign                    ;ESC I = set sign code
008F' 01B3'                             dw      set_noise                    ;ESC J = set noise level
0091' 01BB'                             dw      reset                    ;ESC K = reset simulator
0093' 00B1'                             dw      invalid                    ;ESC L = not used
0095' 00B1'                             dw      invalid                    ;ESC M = not used
0097' 00B1'                             dw      invalid                    ;ESC N = not used
0099' 00B1'                             dw      invalid                    ;ESC O = not used
009B' 00B1'                             dw      invalid                    ;ESC P = not used
009D' 00B1'                             dw      invalid                    ;ESC Q = not used
009F' 00B1'                             dw      invalid                    ;ESC R = not used
00A1' 00B1'                             dw      invalid                    ;ESC S = not used
00A3' 00B1'                             dw      invalid                    ;ESC T = not used
00A5' 00B1'                             dw      invalid                    ;ESC U = not used
00A7' 00B1'                             dw      invalid                    ;ESC V = not used
00A9' 00B1'                             dw      invalid                    ;ESC W = not used
00AB' 00B1'                             dw      invalid                    ;ESC X = not used
00AD' 00B1'                             dw      invalid                    ;ESC Y = not used
00AF' 00B1'                             dw      invalid                    ;ESC Z = not used

00B1'                                invalid:                                ;invalid command, return
00B1' C9                                ret

```

MACRO-80 3.44 09-Dec-81 PAGE 1-3

```

;*
;*      Subroutine :   Get_int
;*      Function :    get integer parameter,
;*                   (signed or unsigned)
;*                   max 32 bits.
;*      Inputs :      none
;*      Outputs :     DEHL = 32 bit parameter
;*
;*****

```

```

00B2'      get_int:
00B2'      C5          push    bc          ;save 8C
00B3'      21 0000     ld      hl,0        ;DEHL = 0
00B6'      11 0000     ld      de,0
00B9'
00B9'      CD 0000*    call    get_byt     ;get byte
00BC'      FE 20       cp      20h
00BE'      38 F9       jr      c.get1     ;ignore control characters
00C0'      FE 40       cp      40h
00C2'      38 0A       jr      c.int_lo   ;low or high byte ?
00C4'      E6 3F       and     00111111b ;high byte : only 6 bits
00C6'      4F          ld      c,a
00C7'      06 06       ld      b,6        ;multiply result by 64 and add new bits
00C9'      CD 00F0*    call    shift
00CC'      18 EB       jr      get1
00CE'
00CE'      F5          push    af          ;low byte
00CF'      E6 0F       and     00001111b ;only 4 bits
00D1'      4F          ld      c,a
00D2'      06 04       ld      b,4        ;multiply result by 16 and add new bits
00D4'      CD 00F0*    call    shift
00D7'      F1          pop     af          ;get sign-bit
00D8'      E6 10       and     10h
00DA'      20 12       jr      nz,int_rdy ;sign-bit set ?

00DC'      7D          ld      a,l        ;invert : DEHL = -DEHL
00DD'      2F          cpl
00DE'      6F          ld      l,a
00DF'      7C          ld      a,h
00E0'      2F          cpl
00E1'      67          ld      h,a
00E2'      7B          ld      a,e
00E3'      2F          cpl
00E4'      5F          ld      e,a
00E5'      7A          ld      a,d
00E6'      2F          cpl
00E7'      57          ld      d,a
00E8'      23          inc     hl
00E9'      7C          ld      a,h
00EA'      85          or      l
00EB'      20 01       jr      nz,int_rdy
00ED'      13          inc     de
00EE'
00EE'      int_rdy:
00EE'      C1          pop     bc
00EF'      C9          ret

```


MACRO-B0 3.44 09-Dec-81 PAGE 1-4

```

00F0'          shift:          ;rotate DEHL B times and add C
00F0' 29          add    hl,hl
00F1' EB          ex     de,hl      ;DEHL = DEHL * 2
00F2' ED 6A       adc    hl,hl
00F4' EB          ex     de,hl
00F5' 10 F9       djnz   shift
00F7' 06 00       ld     b,0        ;add new bits in C
00F9' 09          add    hl,bc
00FA' 00          ret     nc
00FB' 13          inc    de
00FC' C9          ret

```

```

*****
;*
;* Subroutine : Rel_mov          *
;* Function : Relative Id move for all *
;*          stations *
;* Inputs : <int>-parameters in input buffer*
;*          -128..+127 * 25 ns steps *
;* Outputs : Mov_flg's set *
;*          Mov_tim updated *
;*
*****

```

```

00FD'          rel_mov:
00FD' 3A 0000*    ld     a,(nstat##) ;B = station count
0100' 47          ld     b,a
0101' DD 21 0000* ld     ix,mov_flg## ;IX = flag pointer
0105' FD 21 0000* ld     iy,mov_tim## ;IY = pointer to shift time table
0109'          hold:
0109' DD CB 00 46 bit    0,(ix+0)    ;hold if prev. move request not processed yet
010D' 20 FA       jr     nz,hold
010F' CD 00B2'    call   get_int     ;get shift time
0112' FD 75 00    ld     (iy+0),l    ;& store it
0115' DD CB 00 C6 set    0,(ix+0)    ;set move request flag
0119' DD 23       inc    ix          ;move pointers
011B' FD 23       inc    iy
011D' 10 EA       djnz   hold        ;keep count
011F' C9          ret

```

```

*****
;*
;* Subroutine : No_stat          *
;* Function : Set # stations in GRI *
;* Inputs : <int> parameter in buffer *
;*          = # stations *
;* Outputs : Nstat = # stations *
;*
*****

```

macro-80 3.44 09-dec-81 page 1-5

```

0120'          no_stat:
0120'  CD 00B2'          call get_int      ;get parameter
0123'  7D              ld a,l
0124'  32 0000*        ld (nstat#),a      ;& store it

0127'  C9              ret

;*****
;*
;* Subroutine : Set_DTA
;* Function : Set initial dta's
;* Inputs : <int> parameters in buffer.
;*          24 bit in 25 ns units
;* Outputs : dtatab updated
;*
;*****

0128'          set_dta:
0128'  3A 0000*        ld a,(nstat##)      ;B = station count
0128'  47              ld b,a

012C'  DD 21 0000*        ld ix,dtatab##    ;IX = DTA destination pointer
0130'          dta_lp:
0130'  CD 00B2'          call get_int      ;DTA in EHL
0133'  DD 75 00          ld (ix+0),l      ;store it
0136'  DD 74 01          ld (ix+1),h
0139'  DD 73 02          ld (ix+2),e

013C'  DD 23            inc ix            ;move pointer to next DTA
013E'  DD 23            inc ix
0140'  DD 23            inc ix

0142'  10 EC            djnz dta_lp        ;keep count

0144'  C9              ret

;*****
;*
;* Subroutine : Set_sdel
;* Function : Set skywave delays
;* Inputs : <int> parameters in buffer.
;*          16 bit delays in 25 ns units
;* Outputs : Skytab updated
;*
;*****

0145'          set_sdel:
0145'  3A 0000*        ld a,(nstat##)      ;B = station count
0148'  47              ld b,a

0149'  DD 21 0000*        ld ix,skytab##    ;IX = destination pointer
014D'          sdel_lp:
014D'  CD 00B2'          call get_int      ;HL = skywave delay
0150'  DD 75 00          ld (ix+0),l      ;store it
0153'  DD 74 01          ld (ix+1),h

```


MACRO-80 3.44 09-Dec-81 PAGE 1-6

```

0156' DD 23          inc    ix          ;move pointer to next delay
0158' DD 23          inc    ix

015A' 10 F1          djnz   sdel_lp      ;keep count

015C' C9             ret

```

```

;*****
;*
;* Subroutine : Set_glev
;* Function : Set gndwave att. level
;* Inputs : <int> parameters in data buffer
;*          8 bit attenuation in dB's
;* Outputs : Gnd_lev updated
;*
;*****

```

```

015D'          set_glev:
015D' 3A 0000*      ld      a,(nstat##)    ;B - station count
0160' 47           ld      b,a

0161' DD 21 0000*      ld      ix,gnd_lev## ;IX - destination pointer
0165'          glev_lp:
0165' CD 00B2'      call   get_int        ;L = att. level
0168' DD 75 00      ld      (ix+0),l      ;store it

016B' DD 23          inc    ix          ;move pointer

016D' 10 F6          djnz   glev_lp

016F' C9             ret

```

```

;*****
;*
;* Subroutine : Set_slev
;* Function : Set skywave att. level
;* Inputs : <int> parameters in data buffer
;*          8 bit attenuation in dB's
;* Outputs : sky_lev updated
;*
;*****

```

```

0170'          set_slev:
0170' 3A 0000*      ld      a,(nstat##)    ;B - station count
0173' 47           ld      b,a

0174' DD 21 0000*      ld      ix,sky_lev## ;IX - destination pointer
0178'          slev_lp:
0178' CD 00B2'      call   get_int        ;L = att. level
017B' DD 75 00      ld      (ix+0),l      ;store it

017E' DD 23          inc    ix          ;move pointer

0180' 10 F6          djnz   slev_lp

0182' C9             ret

```

MACRO-B0.3.44 09-Dec-81 PAGE 1-7

```

*****
;*
;* Subroutine : Set_gecd
;* Function : Set gndwave ECD's
;* Inputs : <int> parameters in data buffer
;* Outputs : g_ecdtb updated
;*
*****

```

```

0183'      set_gecd:
0183' 3A 0000*      ld    a,(nstat##)    ;B = station count
0186' 47              ld    b,a

0187' DD 21 0000*      ld    ix,g_ecdtb##    ;IX = destination pointer
0188'      set_glp:
0188' CD 00B2'      call get_int    ;L = ECD code
018E' DD 75 00      ld    (ix+0),l    ;store it

0191' DD 23              inc    ix        ;move pointer

0193' 10 F6              djnz   set_glp

0195' C9              ret

```

```

*****
;*
;* Subroutine : Set_secd
;* Function : Set skywave ECD's
;* Inputs : <int> parameters in data buffer
;* Outputs : s_ecdtb updated
;*
*****

```

```

0196'      set_secd:
0196' 3A 0000*      ld    a,(nstat##)    ;B = station count
0199' 47              ld    b,a

019A' DD 21 0000*      ld    ix,s_ecdtb##    ;IX = destination pointer
019E'      set_slp:
019E' CD 00B2'      call get_int    ;L = ECD code
01A1' DD 75 00      ld    (ix+0),l    ;store it

01A4' DD 23              inc    ix        ;move pointer

01A6' 10 F6              djnz   set_slp

01A8' C9              ret

```

```

*****
;*
;* Subroutine : Set_sign
;* Function : Set output polarity
;* Inputs : <int> parameter in data buffer
;*           0 = positive polarity
;*           1 = negative polarity
;*

```


MACRO-B0 3.44 09-Dec-81 PAGE 1-8

```

;*      Outputs :      sign = sign code      *
;*      *                                     *
;*****
01A9'      set_sign:
01A9'      CD 00B2'      call  get_int      :get sign parameter
01AC'      7D           ld      a,1
01AD'      E6 01       and     1           :1 bit only
01AF'      32 0000*    ld      (sign#),a    :store it
01B2'      C9           ret

;*      *                                     *
;*      Subroutine :      Set_noise          *
;*      Function :      Set noise level      *
;*      Inputs :      <int> parameter in data buffer *
;*      Outputs :      noise = noise level    *
;*      *                                     *
;*****

01B3'      set_noise:
01B3'      CD 00B2'      call  get_int      :get noise level
01B6'      7D           ld      a,1
01B7'      32 0000*    ld      (noise#),a    :set level
01BA'      C9           ret

;*      *                                     *
;*      Subroutine :      Reset              *
;*      Function :      Resets simulator      *
;*      Inputs :      none                    *
;*      Outputs :      none                    *
;*      *                                     *
;*****

01BB'      reset:
01BB'      3E 00       ld      a,0          :disable interrupt flip-flop's
01BD'      D3 80       out     (intc1),a
01BF'      D3 81       out     (intc2),a

01C1'      F3           di                :disable interrupts

01C2'      C3 0006'    jp      boot        :start again

01C5'      dseg         :data area
0000*      ds          128              :stack area
0080*      stack:
0080*      00          db      0
end

```

MACRO-80 3.44 09-Dec-81 PAGE 5

Macros:

Symbols:

0006*	BOOT	0045*	BRS_INT	007D*	CMD_TAB
012E*	DTATAB	0130*	DTA_LP	0018	ESC
0065*	ESC_CMD	004D*	EXIT_INT	00B9*	GET1
00BA*	GET_BYT	0053*	GET_CMD	00B2*	GET_INT
0165*	GLEV_LP	0163*	GND_LEV	001D	GS
0189*	G_ECDTB	0109*	HOLD	0007*	INITBRS
0004*	INTRS	0080	INTCL1	0081	INTCL2
00CE*	INT_LO	00EE*	INT_RDY	00B1*	INVALID
0103*	MOV_FLG	0107*	MOV_TIM	01B8*	NOISE
0086	NOISEP	0120*	NO_STAT	0197*	NSTAT
00FD*	REL_MOV	01BB*	RESET	004A*	RX_INT
014D*	SDEL_LP	012B*	SET_DTA	01B3*	SET_GEGD
015D*	SET_GLEV	018B*	SET_GLP	01B3*	SET_NOISE
0145*	SET_SDEL	0196*	SET_SECD	01A9*	SET_SIGN
0170*	SET_SLEV	019E*	SET_SLP	00F0*	SHIFT
01B0*	SIGN	014B*	SKYTAB	0176*	SKY_LEV
0178*	SLEV_LP	0080*	STACK	00A0	SWPORT
019C*	S_ECDTB	0017*	TEST	0091	UARTC

No Fatal error(s)

□

MACRO-80 3.44 09-Dec-81 PAGE 1

```

;*****
;*
;* RS-232 I/O for Loran-C Dynamic Simulator
;*
;* by R.Kellenbach
;*
;* Copyright (C) '86 by the University of Delft
;*
;* Created : nov 1, 1986
;* Last update : dec 10, 1986
;*
;*****

0019      strtcmd equ 19h      ;start command
001C      statcmd equ 1ch     ;buffer status request command
001E      sel1 equ 1eh        ;simulator select codes
001F      sel2 equ 1fh

0090      uartd equ 90h       ;8251 data register
0091      uartc equ 91h       ;8251 control/status register

00A0      swport equ 0a0h     ;dip switch input port

1770      bsize equ 6000      ;host input buffer size
157C      bstop equ 5500      ;handshake thresholds
1388      bstart equ 5000

        .z80

0000*      cseg

cmph1 macro val      ;16 bit compare macro
local cmprdy
ld a,h      ;compare HL with <val>
cp high (val)
jr nz,cmprdy
ld a,l
cp low (val)
cmprdy:
endm

;*****
;*
;* subroutine :   initrs
;* function :     initialize rs-232 interface
;* inputs :       none
;* outputs :      none
;*
;*****

0000*      initrs::
0000* 3E CE      ld a,11001110b ;uart mode instruction
0002* 03 91      out (uartc),a ;8 bits, no parity, 2 stop bits

0004* 3E 27      ld a,00100111b ;enable transmitter & receiver

```


MACRO-80 3.44 09-Dec-81 PAGE 1-1

```

0006' 03 91          out    (uartc),a      ;DTR & RTS high

                                rept    3          ;dummy read's to clear uart
                                in      a,(uartd)
                                endm

0008' 0B 90          +      in      a,(uartd)
000A' 0B 90          +      in      a,(uartd)
000C' 0B 90          +      in      a,(uartd)

000E' 21 0006"       ld      hl,buffer      ;init buffer pointers
0011' 22 0000"       ld      (rdpnt),hl
0014' 22 0002"       ld      (wrpnt),hl

0017' 21 0000       ld      hl,0          ;init byte count
001A' 22 0004"       ld      (bufcnt),hl

                                4
001D' 0B A0          in      a,(swport)      ;read dip switch 8
001F' CB 7F          bit     7,a
0021' 28 04          jr      z,set_2          ;determine select code
0023' 3E 1E          ld      a,sel1
0025' 18 02          jr      sel_ok
0027'               set_2:
0027' 3E 1F          ld      a,sel2
0029'               sel_ok:
0029' 32 1776"       ld      (scode),a

002C' 3E 00          ld      a,0          ;clear select flag
002E' 32 1777"       ld      (selflg),a

0031' C9            ret

;*****
;*
;*      Subroutine :   RX_int
;*      Function :     Uart RX_RDY interrupt
;*                    processing, read character &
;*                    store in buffer
;*      Inputs :       WRPNT = dest. pointer
;*                    BUFCNT = byte count
;*      Outputs :      WRPNT, BUFCNT incremented
;*                    character in buffer
;*
;*
;*****

0032'               rx_int::
0032' 0B 90          in      a,(uartd)      ;get character & clear interrupt request
0034' E6 7F          and     7fh          ;only 7 bits

0036' FE 1E          cp      sel1          ;select code ?
0038' CA 0085"       jp      z,select
003B' FE 1F          cp      sel2
003D' CA 0085"       jp      z,select

0040' 47            ld      b,a          ;save character
0041' 3A 1777"       ld      a,(selflg)    ;simulator selected ?

```

MACRO-80 3.44 09-Dec-81 PAGE 1-2

```

0044' B7          or      a
0045' C8          ret     z          ;no. ignore character & return

0046' 7B          ld      a,b        ;status request ?
0047' FE 1C       cp      statcmd
0049' CA 0098'    jp      z,status

004C' FE 19       cp      strtcmd    ;start command ?
004E' CA 0000*    jp      z,start##

0051' 2A 0004*    ld      hl,(bufcnt) ;get byte count
                                cmphl bsize ;return if buffer full
0054' 7C          +      ld      a,h    ;compare HL with <val>
0055' FE 17       +      cp      high(bsize)
0057' 20 03       +      jr      nz,...0000
0059' 7D          +      ld      a,l
005A' FE 70       +      cp      low(bsize)
005C'             +      ..0000:
005C' C8          ret     z

005D' 23          inc     hl          ;increment buffer count
005E' 22 0004*    ld      (bufcnt),hl

                                cmphl bstop ;buffer almost full ?
0061' 7C          +      ld      a,h    ;compare HL with <val>
0062' FE 15       +      cp      high(bstop)
0064' 20 03       +      jr      nz,...0001
0066' 7D          +      ld      a,l
0067' FE 7C       +      cp      low(bstop)
0069'             +      ..0001:
0069' 20 04       jr      nz,rx_cont ;no. continue

006B' 3E 25       ld      a,00100101b ;handshake : DTR low
006D' D3 91       out     (uartc),a
006F'             rx_cont:
006F' 2A 0002*    ld      hl,(wrpnt)  ;get dest. pointer
0072' 70          ld      (hl),b      ;store character
0073' 23          inc     hl          ;update pointer
                                cmphl buffer+bsize ;past end of buffer ?
0074' 7C          +      ld      a,h    ;compare HL with <val>
0075' FE 17'      +      cp      high(buffer+bsize)
0077' 20 03       +      jr      nz,...0002
0079' 7D          +      ld      a,l
007A' FE 76'      +      cp      low(buffer+bsize)
007C'             +      ..0002:
007C' 20 03       jr      nz,wr_ok
007E' 21 0006*    ld      hl,buffer   ;wrap around
0081'             wr_ok:
0081' 22 0002*    ld      (wrpnt),hl ;save pointer

0084' C9          ret

```

```

*****
;*
;* Subroutine : Select
;* Function : select/deselect simulator
*
```


MACRO-80 3.44 09-Dec-81 PAGE 1-3

```

; *      Inputs :      A = received select code      *
; *      Outputs :     Scode = simulator select code  *
; *      Selflg = select flag                        *
; *
; *****

0085'      select:
0085'      47          ld    b,a          ;compare A with simulator select code
0086'      3A 1776"    ld    a,(scode)
0089'      88          cp    b
008A'      20 06       jr    nz,de_sel    ;deselect simulator if not equal

008C'      3E 01       ld    a,l          ;select simulator
008E'      32 1777"    ld    (selflg),a

0091'      C9          ret

0092'      de_sel:
0092'      3E 00       ld    a,0          ;deselect simulator
0094'      32 1777"    ld    (selflg),a

0097'      C9          ret

; *****
; *
; *      Subroutine :   Status                        *
; *      Function :    Send buffer status to host    *
; *                  computer                          *
; *      Inputs :      Bufcnt = # bytes in buffer    *
; *      Outputs :     none                          *
; *
; *****

0098'      status:
0098'      FB          ei                ;enable burst interrupts while sending status

0099'      2A 0004"    ld    hl,(bufcnt) ;get buffer byte count

009C'      7D          ld    a,l          ;& send it
009D'      CD 00E6"    call send
00A0'      7C          ld    a,h
00A1'      CD 00E6"    call send

00A4'      C9          ret

; *****
; *
; *      Subroutine :   Buf_read                      *
; *      Function :    Read character from input     *
; *                  buffer                          *
; *      Inputs :      RDPNT = source pointer        *
; *                  BUFCNT = byte count            *
; *      Outputs :     A = character from buffer     *
; *
; *****

```

MACRO-80 3.44 09-Dec-81 PAGE 1-4

```

00A5'          buf_read:
00A5'  E5          push    hl
00A6'  2A 0000"    ld      hl,(rdpnt)    ;get pointer
00A9'  7E          ld      a,(hl)        ;get character
00AA'  F5          push    af            ;& save it

00AB'  23          inc     hl            ;update pointer
                                cmphl    buffer+bsize    ;past end of buffer ?
00AC'  7C          +      ld      a,h      ;compare HL with <val>
00AD'  FE 17"      +      cp      high (buffer+bsize)
00AF'  20 03      +      jr      nz,..0003
00B1'  7D          +      ld      a,l
00B2'  FE 76"      +      cp      low (buffer+bsize)
00B4'          +      ..0003:
00B4'  20 03      jr      nz,rd_ok        ;no..continue
00B6'  21 0006"    ld      hl,buffer      ;wrap around
00B9'          rd_ok:
00B9'  22 0000"    ld      (rdpnt),hl    ;save pointer

00BC'  F3          di                    ;no interrupts here
00BD'  2A 0004"    ld      hl,(bufcnt)    ;decrement byte count
00C0'  2B          dec     hl
00C1'  22 0004"    ld      (bufcnt),hl
00C4'  FB          ei

                                cmphl    bstart        ;lower threshold reached ?
00C5'  7C          +      ld      a,h      ;compare HL with <val>
00C6'  FE 13      +      cp      high (bstart)
00C8'  20 03      +      jr      nz,..0004
00CA'  7D          +      ld      a,l
00CB'  FE 88      +      cp      low (bstart)
00CD'          +      ..0004:
00CD'  20 04      jr      nz,not_en

00CF'  3E 27      ld      a,00100111b    ;enable host transmitter : DTR high
00D1'  D3 91      out     (uartc),a
00D3'          not_en:
00D3'  F1          pop     af            ;restore character & return
00D4'  E1          pop     hl
00D5'  C9          ret

;*****
;*
;* Subroutine :    RX_stat
;* Function :      get input buffer status
;* Inputs :        BUFCNT = byte count
;* Outputs :       zero flag : NZ if character
;*                 available in buffer
;*
;*****

00D6'          rx_stat::
00D6'  E5          push    hl
00D7'  2A 0004"    ld      hl,(bufcnt)    ;get byte count
00DA'  7C          ld      a,h
00DB'  B5          or      l            ;set zero flag

```


MACRO-80 3.44 09-Dec-81 PAGE 1-5

```

00DC' E1                pop    hl
00DD' C9                ret

;*****
;*
;* Subroutine :   Get_byt
;* Function :    get character from buffer
;* Inputs :      BUFCNT = byte count
;*               data in buffer
;* Outputs :     A = character
;*
;*****

00DE'                get_byt::
00DE' CD 00D6'         call    rx_stat      ;get buffer status
00E1' 28 FB           jr      z,get_byt     ;wait until character available
00E3' C3 00A5'         jp      buf_read    ;get character & return

;*****
;*
;* subroutine :   Send
;* function :     send character to host computer
;* inputs :      a = character
;* outputs :     none
;*
;*****

00E6'                send::
00E6' F5              push    af
00E7'                sendlp:
00E7' DB 91           in      a,(uartc)     ;uart ready ?
00E9' E6 01           and     1
00EB' 28 FA           jr      z,sendlp

00ED' F1              pop     af
00EE' D3 90           out     (uartd),a     ;send character

00F0' C9              ret

;***** data area *****

00F1'                dseg

0000"                rdpnt: ds    2          ;buffer read pointer
0002"                wrpnt: ds    2          ;buffer write pointer
0004"                bufcnt: ds    2          ;buffer byte count
0006"                buffer: ds    bsize      ;data buffer
1776"                scode: ds    1          ;simulator select code
1777"                selflg: ds    1          ;simulator select flag

1778" 00              db      0
                        end

```


MACRO-80 3.44 09-Dec-81 PAGE 5

Macros:

CMPHL

Symbols:

005C' ..0000	0069' ..0001	007C' ..0002
0084' ..0003	00CD' ..0004	1770 BSIZE
1388 BSTART	157C BSTOP	0004' BUF CNT
0006' BUFFER	00A5' BUF_READ	0092' DE_SEL
000E1' GET_BYT	0000I' INITRS	0003' NOT_EN
0000' RDPNT	0089' RD_OK	006F' RX_CONT
0032I' RX_INT	0006I' RX_STAT	1776' SCODE
001E SEL1	001F SEL2	0085' SELECT
1777' SELFLG	0029' SEL_OK	00E6I' SEND
00E7' SENDLP	0027' SET_2	004F' START
001C STATCMD	0098' STATUS	0019 STRTCMD
00A0 SWPORT	0091 UARTC	0090 UARTD
0002' WRPNT	0081' WR_OK	

No Fatal error(s)

□

MACRO-80 3.44 09-Dec-81

PAGE 1

```

*****
;*
;*      Loran-C Dynamic Simulator Burst Interrupt
;*      Processing Routines
;*
;*      by R.Kellenbach
;*
;*      Copyright (C) '86 by the University of Delft
;*
;*      Created      : dec  3, 1986
;*      Last update  : jan 27, 1988
;*
*****

0000      timer0 equ  00h      :8253 timers
0001      timer1 equ  01h
0002      timer2 equ  02h
0003      timmod equ  03h      :8253 mode register

0004      gnddel equ  04h      :gndwave delay port
0008      skydel equ  08h      :skywave delay port

000C      delport equ  0ch      :delay setting port (fine delay steps)

0010      gndecd equ  10h      :gndwave ECD port
0014      skyecd equ  14h      :skywave ECD port

0018      dacdata equ  18h      :LOGDAC data port
001C      daccont equ  1ch      :LOGDAC control port

0080      intc11 equ  80h      :interrupt clear flip-flop's
0081      intc12 equ  81h

0082      trig1 equ  82h      :scope trigger ports
0083      trig2 equ  83h
0084      trig3 equ  84h

      .z80

0000*      cseg

cmph1 macro val      :16 bit compare macro
      local cmprdy
      ld  a,h      :compare HL with <val>
      cp  high (val)
      jr  nz,cmprdy
      ld  a,l
      cp  low (val)
cmprdy:
      endm

*****
;*
;*      Subroutine :   Initbrs
;*      Function  :   Initialize variables and
;*                   hardware for burst interrupts

```

MACRO-80 3.44 09-Dec-81

PAGE 1-1

```

;*      Inputs :      none                      *
;*      Outputs :     misc. variables initialized *
;*                                                    *
;*****
0000'      initbrs::
0000'      3E 34          ld      a,00110100b      ;init timer 0
0002'      D3 03          out     (timmod),a        ;mode 2 (rate generator)
0004'      3E 32          ld      a,50             ;generate 100 kHz pulse
0006'      D3 00          out     (timer0),a
0008'      3E 00          ld      a,0
000A'      D3 00          out     (timer0),a

000C'      3E 74          ld      a,01110100b      ;init timer 1
000E'      D3 03          out     (timmod),a        ;mode 2 (rate generator)

0010'      3E BA          ld      a,10111010b      ;init timer 2
0012'      D3 03          out     (timmod),a        ;mode 5 (hardware triggered strobe)

0014'      3E 00          ld      a,0              ;disable interrupt flip-flop's
0016'      D3 80          out     (intc11),a
0018'      D3 81          out     (intc12),a

001A'      3E 00          ld      a,0              ;select prom area for ECD
001C'      D3 10          out     (gndec1),a
001E'      D3 14          out     (skyecd),a

0020'      3E 00          ld      a,0              ;init delay PAL's
0022'      D3 04          out     (gnddl),a
0024'      D3 08          out     (skydel),a

0026'      3E 0F          ld      a,00001111b      ;init LOGDAC control port, disable DAC's
0028'      D3 1C          out     (daccont),a

002A'      3E 00          ld      a,0              ;enable trig3 pulses at every interrupt
002C'      D3 84          out     (trig3),a

002E'      3E 00          ld      a,0              ;index = 0
0030'      32 0000'      ld      (index),a

0033'      3E 07          ld      a,7              ;brsct = 7
0035'      32 0002'      ld      (brsct),a

0038'      3E 00          ld      a,0              ;set sign bit
003A'      32 000F'      ld      (sign),a

003D'      3E 03          ld      a,3              ;3 stations
003F'      32 0001'      ld      (nstat),a

0042'      21 0000          ld      hl,0            ;remainder = 0
0045'      22 000B'      ld      (remain),hl

0048'      21 00AC'      ld      hl,phcinit        ;init phase codes
004B'      11 0004'      ld      de,phctab
004E'      01 0007          ld      bc,7
0051'      ED 80          ldir

```


MACRO-80 3.44 09-Dec-81

PAGE 1-2

```

0053' 3A 0004'      1d      a,(phctab)      ;phase code = 1st master code
0056' 32 0003'      1d      (phcode),a

0059' 21 00B3'      1d      hl,dtainit      ;init dta's
005C' 11 0011'      1d      de,dtatab
005F' 01 0015      1d      bc,3*7
0062' ED B0        1dir

0064' 21 00C8'      1d      hl,skyinit      ;init skywave delay's
0067' 11 0026'      1d      de,skytab
006A' 01 000E      1d      bc,2*7
006D' ED B0        1dir

006F' 21 00D6'      1d      hl,g_levinit     ;init gndwave levels
0072' 11 0034'      1d      de,gnd_lev
0075' 01 0007      1d      bc,7
0078' ED B0        1dir

007A' 21 00DD'      1d      hl,s_levinit     ;init skywave levels
007D' 11 003B'      1d      de,sky_lev
0080' 01 0007      1d      bc,7
0083' ED B0        1dir

0085' 21 00E4'      1d      hl,ecdinit      ;init gndwave ECD's
0088' 11 0042'      1d      de,g_ecdtb
008B' 01 0007      1d      bc,7
008E' ED B0        1dir

0090' 21 00F4'      1d      hl,ecdinit      ;init skywave ECD's
0093' 11 0049'      1d      de,s_ecdtb
0096' 01 0007      1d      bc,7
0099' ED B0        1dir

009B' 21 00EB'      1d      hl,mov_init      ;clear move flags
009E' 11 0050'      1d      de,mov_flg
00A1' 01 0007      1d      bc,7
00A4' ED B0        1dir

00A6' 3E FF        1d      a,255            ;no noise
00AB' 32 0010'      1d      (noise),a

00AB' C9           ret

00AC' CA F9 F9 F9      phcinit:db 0cah,0f9h,0f9h,0f9h,0f9h,0f9h,0f9h ;initial phase codes
00B0' F9 F9 F9

00B3' A0 86 01      dtainit:db 160,134,1      ;2.5 ms
00B6' C0 D4 01      db 192,212,1      ;3.0 ms
00B9' E0 22 02      db 224,34,2      ;3.5 ms

rept 4
db 0,0,0
endm

00BC' 00 00 00      +      db 0,0,0
00BF' 00 00 00      +      db 0,0,0
00C2' 00 00 00      +      db 0,0,0

```

MACRO-80 3.44 09-Dec-81 PAGE 1-3

```

00C5' 00 00 00      +          db      0,0,0

00C8' 0320 0384          skyinit:dw      800,900,1000,0;0,0,0      ;20, 22.5, 25 us
00CC' 03E8 0000
00D0' 0000 0000
00D4' 0000

00D6' 00 10 35 00          g_levinit:db      0,16,53,0,0,0,0      ;0, -6, -20 dB
00DA' 00 00 00
00DD' FF FF FF FF          s_levinit:db      255,255,255,255,255,255      ;no skywaves
00E1' FF FF FF

00E4' 0A 0A 0A 0A          ecdinit:db      10,10,10,10,10,10      ;initial ECD's : 0 us
00E8' 0A 0A 0A

00EB' 00 00 00 00          mov_init:db      0,0,0,0,0,0,0
00EF' 00 00 00

```

```

;*****
;*
;*      Subroutine :      Start
;*      Function :      start burst generating process
;*      Inputs :      none
;*      Outputs :      none
;*
;*****

```

```

00F2'          start::
00F2' 3E 64          ld      a,100      ;set timer 1 for 1 ms interval
00F4' D3 01          out     (timer1),a
00F6' 3E 00          ld      a,0
00F8' D3 01          out     (timer1),a      ;timer is running now...

00FA' 3E 00          ld      a,0      ;clear interrupt flip-flop 1
00FC' D3 80          out     (intc1),a
00FE' 3E 01          ld      a,1      ;enable interrupts
0100' D3 80          out     (intc1),a

0102' C9          ret

```

```

;*****
;*
;*      Subroutine :      Brsint
;*      Function :      Burst timer interrupt processing*
;*      Inputs :      misc. variables
;*      Outputs :      misc. variables
;*
;*****

```

```

0103'          brs_int::
0103' 3E 00          ld      a,0      ;clear interrupt flip-flop
0105' D3 80          out     (intc1),a
0107' 3E 01          ld      a,1
0109' D3 80          out     (intc1),a

010B' FB          ei      ;enable RS-232 interrupts

```


MACRO-80 3.44 09-Dec-81 PAGE 1-4

```

010C' 3E 01          ld    a,l          ;disable trigger 1 & 2 pulses
010E' D3 82          out    (trig1).a
0110' D3 83          out    (trig2).a

0112' 3A 0000"       ld    a,(index)    ;bc = station index
0115' 4F             ld    c,a
0116' 06 00          ld    b,0

0118' 21 0002"       ld    hl,brsct     ;decrement burst counter
011B' 35             dec    (hl)

011C' FA 015C'       jp     m,next_stat ;next station ?

011F' CC 0126'       call    z,next_dta  ;prepare for next dta ?

0122' CD 01B3'       call    set_phc     ;set phase code bits

0125' C9             ret

```

```

*****
,*                                     *
,*      Subroutine :   next_dta        *
,*      Function :    prepare timer for next dta    *
,*      Inputs :      dta in dta table    *
,*      Outputs :     remain = remainder fraction    *
,*                   for delay lines    *
,*                                     *
*****

```

```

0126'                next_dta:
0126' C5             push    bc

0127' 21 0011"       ld    hl,datab     ;HLDE = dta[i]
012A' 09             add    hl,bc
012B' 09             add    hl,bc
012C' 09             add    hl,bc
012D' 5E             ld    e,(hl)
012E' 23             inc    hl
012F' 56             ld    d,(hl)
0130' 23             inc    hl
0131' 6E             ld    l,(hl)
0132' 26 00          ld    h,0

0134' 01 0190        ld    bc,400       ;convert 25 ns units to 10 us steps
0137' CD 0324'       call    div32

013A' ED 4B 000B"    ld    bc,(remain)  ;add new remainder
013E' 09             add    hl,bc

                                cmpl    400       ;remainder >= 10 us ?
013F' 7C             +      ld    a,h          ;compare HL with <val>
0140' FE 01          +      cp     high (400)
0142' 20 03          +      jr     nz,...0000
0144' 7D             +      ld    a,l
0145' FE 90          +      cp     low (400)

```

MACRO-80 3.44 09-Dec-81 PAGE 1-5

```

0147'      +      ..0000:
0147'  38 05      jr      c.rm_ok

0149'  01 FE70      ld      bc,-400      ;overflow : correct HL
014C'  09          add     hl,bc
014D'  13          inc     de      ;add 10 us step for timer
014E'      rm_ok:
014E'  22 000B*     ld      (remain),hl      ;save remainder for delay lines
0151'  78          ld      a,e      ;set timer for next dta
0152'  D3 01      out     (timer1),a
0154'  7A          ld      a,d
0155'  D3 01      out     (timer1),a

0157'  C1          pop     bc

0158'  CD 02D3*     call    chk_mov      ;move request ?

0158'  C9          ret

;*****
;*
;*      Subroutine :      Next_stat      *
;*      Function :      prepare burst generator for      *
;*                      next station      *
;*      Inputs :      misc. variables      *
;*      Outputs :      misc. variables      *
;*
;*****

015C'      next_stat:
015C'  21 00D0      ld      hl,20B      ;1 ms delay to finish current burst
015F'      delay:
015F'  2B          dec     hl
0160'  7C          ld      a,h
0161'  B5          or      l
0162'  C2 015F*     jp      nz,delay

0165'  3E 07      ld      a,7      ;burst counter = 7
0167'  32 0002*     ld      (brsct),a

016A'  3E 64      ld      a,100      ;prepare timer for 1 ms interval
016C'  D3 01      out     (timer1),a
016E'  3E 00      ld      a,0
0170'  D3 01      out     (timer1),a

0172'  21 0004*     ld      hl,phctab      ;invert previous phase code for next GRI
0175'  09          add     hl,bc
0176'  7E          ld      a,(hl)
0177'  EE 55      xor      01010101b
0179'  77          ld      (hl),a

017A'  3A 0000*     ld      a,(index)      ;increment station index
017D'  3C          inc     a
017E'  21 0001*     ld      hl,nstat
0181'  BE          cp      (hl)      ;last station ?
0182'  38 06      jr      c,index_ok

```


MACRO-80 3.44 09-Dec-81 PAGE 1-6

```

0184' 3E 00          ld    a,0          ;generate trig1 pulse
0186' D3 82          out   (trig1),a

0188' 3E 00          ld    a,0          ;wrap around
018A'                index_ok:
018A' 32 0000"       ld    (index),a

018D' 4F             ld    c,a          ;bc = next index
018E' 06 00          ld    b,0

0190' 21 0004"       ld    hl,phctab    ;select new phase code
0193' 09             add    hl,bc
0194' 7E             ld    a,(hl)
0195' 32 0003"       ld    (phcode),a

0198' 3E 00          ld    a,0          ;generate trig2 pulse
019A' D3 83          out   (trig2),a

019C' 21 0042"       ld    hl,g_ecdtb    ;copy gndwave ECD[i]
019F' 09             add    hl,bc
01A0' 7E             ld    a,(hl);
01A1' 32 0000"       ld    (g_ecd),a

01A4' 21 0049"       ld    hl,s_ecdtb    ;copy skywave ECD[i]
01A7' 09             add    hl,bc
01A8' 7E             ld    a,(hl)
01A9' 32 000E"       ld    (s_ecd),a

01AC' CD 01E0"       call   set_del      ;set gndwave & skywave delay ports
01AF' CD 028A"       call   set_lev      ;set signal levels
01B2' C9             ret

```

```

;*****
;*
;* Subroutine : Set_phc
;* Function : Set phase code control bits
;*           & ECD for gndwave & skywave
;* Inputs : g_ecd = gndwave ECD cont. bits
;*          s_ecd = skywave ECD cont. bits
;*          phcode = phase code
;* Outputs : phcode = rotated phase code
;*
;*****

```

```

01B3'                set_phc:
01B3' 3A 0003"       ld    a,(phcode)    ;rotate phase code
01B6' 17             rla
01B7' 32 0003"       ld    (phcode),a

01BA' 17             rla                ;shift phase code to bit 0
01BB' C5             push    bc
01BC' 47             ld    b,a
01BD' 3A 000F"       ld    a,(sign)      ;xor phase code bit with sign bit

```


MACRO-B0 3.44 09-Dec-81 PAGE 1-7

```

01C0*  A8                xor    b
01C1*  C1                pop    bc

01C2*  CB 47             bit     0,a
01C4*  20 0B             jr      nz,set_bit      ; + or - code ?

01C6*  3A 000D*          ld      a,(g_ecd)        ;output gndwave ECD control bits
01C9*  D3 10             out     (gndecd),a

01CB*  3A 000E*          ld      a,(s_ecd)        ;output skywave ECD control bits
01CE*  D3 14             out     (skyecd),a

01D0*  C9                ret

01D1*                      set_bit:
01D1*  3A 000D*          ld      a,(g_ecd)        ;output gndwave ECD control bits
01D4*  CB EF             set     5,a             ;set sign bit
01D6*  D3 10             out     (gndecd),a

01D8*  3A 000E*          ld      a,(s_ecd)        ;output skywave ECD control bits
01DB*  CB EF             set     5,a             ;set sign bit
01DD*  D3 14             out     (skyecd),a

01DF*  C9                ret

```

```

*****
;*                               *
;* Subroutine : Set_del          *
;* Function : set gndwave & skywave delay *
;*           ports              *
;* Inputs : remain = gndwave DTA fraction *
;*           data in skytab      *
;*           BC = index for next station *
;* Outputs : none                *
;*                               *
*****

```

```

01E0*                      set_del:
01E0*  2A 000B*          ld      hl,(remain)      ;get remainder of DTA
                                rept    3          ;convert to 200 ns units
                                srl     h
                                rr      l          ;divide HL by 8
                                endm
01E3*  CB 3C             +      srl     h
01E5*  CB 1D             +      rr      l          ;divide HL by 8
01E7*  CB 3C             +      srl     h
01E9*  CB 1D             +      rr      l          ;divide HL by 8
01EB*  CB 3C             +      srl     h
01ED*  CB 1D             +      rr      l          ;divide HL by 8

01EF*  3E 40             ld      a,01000000b      ;switch carriers off
01F1*  D3 10             out     (gndecd),a
01F3*  D3 14             out     (skyecd),a

01F5*  3E 31             ld      a,49             ;set gndwave delay PAL
01F7*  95                sub     1

```

MACRO-80 3.44 09-Dec-81

PAGE 1-8

```

01FB' 03 04          out    (gnddel),a

01FA' 21 0026*       ld      hl,skytb      ;DE = skywave delay
01FD' 09             add     hl,bc
01FE' 09             add     hl,bc
01FF' 5E             ld      e,(hl)
0200' 23             inc     hl
0201' 56             ld      d,(hl)

0202' 2A 000B*       ld      hl,(remain)    ;add remainder
0205' 19             add     hl,de
0206' EB             ex      de,hl

0207' 21 0000       ld      hl,0           ;convert to 10 us units
020A' C5             push    bc
020B' 01 0190       ld      bc,400
020E' CD 0324*      call    div32
0211' C1             pop     bc

0212' 1B             dec     de             ;decrement timer count
                                           ;(timer needs 1 clock pulse to trigger)

0213' 7B             ld      a,e           ;set skywave delay timer
0214' D3 02          out     (timer2),a
0216' 7A             ld      a,d
0217' D3 02          out     (timer2),a

0219' E5             push    hl           ;save skywave delay fraction
                                           rept 3
                                           srl     h           ;convert to 200 ns units
                                           rr      l
                                           endm

021A' CB 3C          +       srl     h           ;convert to 200 ns units
021C' CB 1D          +       rr      l
021E' CB 3C          +       srl     h           ;convert to 200 ns units
0220' CB 1D          +       rr      l
0222' CB 3C          +       srl     h           ;convert to 200 ns units
0224' CB 1D          +       rr      l

0226' 3E 31          ld      a,49          ;set skywave delay PAL
0228' 95             sub     l
0229' D3 08          out     (skydel),a

022B' E1             pop     hl
022C' 7D             ld      a,l           ;get 25 ns units of skywave delay
022D' E6 07          and     7
022F' CB 27          sla     a           ;shift into position
0231' CB 27          sla     a
0233' CB 27          sla     a
0235' 5F             ld      e,a

0236' 2A 000B*       ld      hl,(remain)    ;get 25 units of gndwave delay
0239' 7D             ld      a,l
023A' E6 07          and     7
023C' B3             or      e           ;add skywave bits
023D' D3 0C          out     (delport),a    ;set delay lines

```


MACRO-80 3.44 09-Dec-81 PAGE 1-9

```

023F' 3E B2          ld      a,178          :delay 500 us to suppress LPF filter
0241'               wait:                :transients due to phase modulation
0241' 3D             dec      a
0242' C2 0241'       jp      nz,wait

0245' 3A 0003"       ld      a,(phcode)      :rotate phase code
0248' 17             rla
0249' 32 0003"       ld      (phcode),a

024C' 17             rla                    :shift phase code to bit 0
024D' C5             push    bc
024E' 47             ld      b,a
024F' 3A 000F"       ld      a,(sign)        :xor phase code bit with sign bit
0252' A8             xor      b
0253' C1             pop     bc

0254' CB 47         bit      0,a
0256' 20 17         jr      nz,set_b1        ; + or - code ?

0258' 3A 000D"       ld      a,(g_ecd)       :output gndwave ECD control bits
025B' CB F7         set      6,a
025D' D3 10         out      (gndecd),a
025F' CB B7         res      6,a             :enable carrier
0261' D3 10         out      (gndecd),a

0263' 3A 000E"       ld      a,(s_ecd)       :output skywave ECD control bits
0266' CB F7         set      6,a
0268' D3 14         out      (skyecd),a
026A' CB B7         res      6,a             :enable carrier
026C' D3 14         out      (skyecd),a

026E' C9             ret

026F'               set_b1:
026F' 3A 000D"       ld      a,(g_ecd)       :output gndwave ECD control bits
0272' CB EF         set      5,a             :set sign bit
0274' CB F7         set      6,a
0276' D3 10         out      (gndecd),a
0278' CB B7         res      6,a             :enable carrier
027A' D3 10         out      (gndecd),a

027C' 3A 000E"       ld      a,(s_ecd)       :output skywave ECD control bits
027F' CB EF         set      5,a             :set sign bit
0281' CB F7         set      6,a
0283' D3 14         out      (skyecd),a
0285' CB B7         res      6,a             :enable carrier
0287' D3 14         out      (skyecd),a

0289' C9             ret

```

```

*****
;*
;* Subroutine : Set_lev
;* Function : set gndwave & skywave signal
;*           levels for next station
;*

```

MACRO-80 3.44 09-Dec-81 PAGE 1-10

```

;*      Inputs :      data in gnd_lev & sky_lev      *
;*                                     tables          *
;*                                     BC = index for next station *
;*      Outputs :      none                          *
;*                                     *
;*****

```

```

028A'      set_lev:
028A'      1E 30          ld      e,0110000b      ;40 dB att. control bits

028C'      21 0034*      ld      h1,gnd_lev      ;send gndwave level to LOGDAC data port
028F'      09           add     h1,bc
0290'      7E           ld      a,(h1)
0291'      FE 69        cp      105              ;40 dB att. on or off ?
0293'      38 04        jr      c,glev_ok
0295'      D6 69        sub     105
0297'      CB A3        res     4,e
0299'      glev_ok:
0299'      03 18        out     (dacdata),a

029B'      3E 07        ld      a,00000111b      ;enable DAC CS-lines
029D'      03 1C        out     (daccont),a
029F'      CB 87        res     0,a              ;generate WR-pulse for gndwave DAC
02A1'      D3 1C        out     (daccont),a
02A3'      CB C7        set     0,a
02A5'      D3 1C        out     (daccont),a

02A7'      21 003B*      ld      h1,sky_lev      ;send skywave level to LOGDAC data port
02AA'      09           add     h1,bc
02AB'      7E           ld      a,(h1)
02AC'      FE 69        cp      105              ;40 dB att. on or off ?
02AE'      38 04        jr      c,slev_ok
02B0'      D6 69        sub     105
02B2'      CB AB        res     5,e
02B4'      slev_ok:
02B4'      D3 18        out     (dacdata),a

02B6'      3E 05        ld      a,00000101b      ;generate WR-pulse for skywave DAC
02B8'      D3 1C        out     (daccont),a
02BA'      CB CF        set     1,a
02BC'      D3 1C        out     (daccont),a

02BE'      3A 0010*      ld      a,(noise)       ;set noise level
02C1'      D3 18        out     (dacdata),a
02C3'      3E 03        ld      a,00000011b      ;generate WR-pulse for noise DAC
02C5'      D3 1C        out     (daccont),a
02C7'      CB D7        set     2,a
02C9'      D3 1C        out     (daccont),a

02CB'      CB DF        set     3,a              ;disable DAC CS-lines
02CD'      D3 1C        out     (daccont),a

02CF'      83           or      e                ;set 40 dB att. control bits
02D0'      D3 1C        out     (daccont),a

02D2'      C9           ret

```


MACRO-80 3.44 09-Dec-81 PAGE 1-11

```

;*****
;*
;*      Subroutine :   Chk_mov
;*      Function :   Process external move request
;*                  (if any)
;*      Inputs :     Mov_flg = move request flag
;*                  Mov_tim = shift time
;*      Outputs :    DTA's modified
;*
;*****

02D3'      chk_mov:
02D3'      21 0050*      ld      hl,mov_flg      ;get mov_flg for current station
02D6'      09           add     hl,bc
02D7'      7E           ld      a,(hl)
02D8'      B7           or      a
02D9'      C8           ret     z      ;return if no request

02DA'      36 00       ld      (hl),0      ;clear flag

02DC'      C5           push    bc

02DD'      21 0057*      ld      hl,mov_tim      ;E = shift time
02E0'      09           add     hl,bc
02E1'      5E           ld      e,(hl)

02E2'      21 0011*      ld      hl,datab      ;dta(i) = dta(i) - shift time
02E5'      09           add     hl,bc
02E6'      09           add     hl,bc
02E7'      09           add     hl,bc
02E8'      CD 031A'      call    sgn_ext
02EB'      7E           ld      a,(hl)
02EC'      93           sub     e
02ED'      77           ld      (hl),a
02EE'      23           inc     hl
02EF'      7E           ld      a,(hl)
02F0'      9A           sbc     a,d
02F1'      77           ld      (hl),a
02F2'      23           inc     hl
02F3'      7E           ld      a,(hl)
02F4'      99           sbc     a,c
02F5'      77           ld      (hl),a

02F6'      3A 0000*      ld      a,(index)      ;get prev. index
02F9'      3D           dec     a
02FA'      F2 0301'      jp      p,not_arnd
02FD'      3A 0001*      ld      a,(nstat)      ;wrap around if master
0300'      3D           dec     a
0301'      not_arnd:
0301'      4F           ld      c,a
0302'      06 00       ld      b,0
0304'      21 0011*      ld      hl,datab      ;dta (i-1) = dta (i-1) + shift time
0307'      09           add     hl,bc
0308'      09           add     hl,bc
0309'      09           add     hl,bc

```

MACRO-80 3.44 09-Dec-81 PAGE 1-12

```

030A' 0D 031A'      call    sgn_ext
030D' 7E            ld      a,(hl)
030E' 83            add     a,e
030F' 77            ld      (hl),a
0310' 23            inc     hl
0311' 7E            ld      a,(hl)
0312' 8A            adc     a,d
0313' 77            ld      (hl),a
0314' 23            inc     hl
0315' 7E            ld      a,(hl)
0316' 89            adc     a,c
0317' 77            ld      (hl),a

```

```

0318' C1            pop     bc

```

```

0319' C9            ret

```

```

031A'              sgn_ext:              ;sign_extend of E to CDE

```

```

031A' 0F 00          ld      c,0
031C' 16 00          ld      d,0
031E' 7B            ld      a,e
031F' 17            rla
0320' D0            ret     nc
0321' 0D            dec     c
0322' 15            dec     d
0323' C9            ret

```

```

*****
;*
;*      Subroutine :   Div32
;*      Function :    unsigned 32 bit division
;*      Inputs :      HLDE = 32 bit dividend
;*                   BC = 16 bit divisor
;*      Outputs :     DE = quotient
;*                   HL = remainder
;*
*****

```

```

0324'              div32:
0324' 3E 10          ld      a,16          ;bit count
0326'              divlp:
0326' 29            add     hl,hl          ;HLDE = HLDE * 2
0327' EB            ex      de,hl
0328' 29            add     hl,hl
0329' EB            ex      de,hl
032A' D2 032E'      jp      nc,hl_ok
032D' 23            inc     hl
032E'              hl_ok:
032E' 87            or      a              ;compare BC with 16 MSB's of HLDE
032F' ED 42          sbc     hl,bc
0331' DA 0338'      jp      c,not_ok
0334' 13            inc     de              ;increment quotient
0335' C3 0339'      jp      next
0338'              not_ok:

```

MACRO-B0 3.44 09-Dec-81 PAGE 1-13

```

0338' 09          add    hl,bc      ;correct HLDE
0339'          next:
0339' 3D          dec     a          ;keep count
033A' C2 0326'    jp      nz,divlp
033D' C9          ret

;***** data area *****

033E'          dseg

0000"          index: ds    1      ;station index
0001"          nstat:: ds    1      ;total # stations
0002"          brsct: ds    1      ;burst counter
0003"          phcode: ds    1      ;current phase code
0004"          phctab: ds    7      ;phase code table
000B"          remain: ds    2      ;dta fraction in 25 ns units
000D"          g_ecd:  ds    1      ;gndwave ECD control bits
000E"          s_ecd:  ds    1      ;skywave ECD control bits
000F"          sign::  ds    1      ;external controlled sign bit
0010"          noise:: ds    1      ;noise level

0011"          dtatab:: ds   3*7    ;dta data table, 24 bit entries in 25 ns units
0026"          skytab:: ds   2*7    ;skywave delay data table, 16 bit, 25 ns units
0034"          gnd_lev::ds    7      ;gndwave signal levels
0038"          sky_lev::ds    7      ;skywave signal levels
0042"          g_ecdtb::ds    7      ;gndwave ECD data table
0049"          s_ecdtb::ds    7      ;skywave ECD data table

0050"          mov_flg::ds    7      ;move request flags
0057"          mov_tim::ds    7      ;shift time

005E" 00          db      0
          end

```


MACRO-80 3.44 09-Dec-81 PAGE 5

Macros:

CMPHL

Symbols:

0147*	..0000	0002*	BRSCT	01031*	BRS_INT
0203*	CHK_MOV	001C	DACCONT	0018	DACDATA
015F*	DELAY	000C	DELPORT	0324*	DIV32
0326*	DIVLP	00B3*	DTAINIT	00111*	DTATAB
00E4*	ECDINIT	0299*	GLEV_OK	0004	GNODEL
0010	GNOECD	00341*	GNO_LEV	0000*	G_ECD
00421*	G_ECDTB	00D6*	G_LEVINIT	032E*	HL_OK
0000*	INDEX	018A*	INDEX_OK	00001*	INITBRS
0080	INTCL1	0081	INTCL2	00501*	MOV_FLG
00E8*	MOV_INIT	00571*	MOV_TIM	0339*	NEXT
0126*	NEXT_DTA	015C*	NEXT_STAT	00101*	NOISE
0301*	NOT_ARND	0338*	NOT_OK	00011*	NSTAT
00AC*	PHCINIT	0003*	PHCODE	0004*	PHCTAB
0008*	REMAIN	014E*	RM_OK	026F*	SET_B1
01D1*	SET_BIT	01E0*	SET_DEL	028A*	SET_LEV
01B3*	SET_PHC	031A*	SGN_EXT	000F1*	SIGN
0008	SKYDEL	0014	SKYECD	00C8*	SKYINIT
00261*	SKYTAB	003B1*	SKY_LEV	0284*	SLEV_OK
00F21*	START	000E*	S_ECD	00491*	S_ECDTB
00DD*	S_LEVINIT	0000	TIMER0	0001	TIMER1
0002	TIMER2	0003	TIMMOD	0082	TRIG1
0083	TRIG2	0084	TRIG3	0241*	WAIT

No Fatal error(s).

□

MACRO-80 3.44 09-Dec-81 PAGE 1

```

*****
;*
;*   Loran-C Dynamic Simulator Hardware Testing
;*   Routines
;*
;*   by R.Kellénbach
;*
;*   Copyright (C) '86 by the University of Delft
;*
;*   Created   : dec  4, 1986
;*   Last update : april 30, 1987
;*
*****

0000          timer0 equ 00h          :8253 timers
0001          timer1 equ 01h
0002          timer2 equ 02h
0003          timmod equ 03h          :8253 mode register

0004          gnddel equ 04h          :gndwave delay port
0008          skydel equ 08h          :skywave delay port

000C          delport equ 0ch          :delay setting port (fine delay steps)

0010          gndecd equ 10h          :gndwave ECD port
0014          skyecd equ 14h          :skywave ECD port

0018          dacdata equ 18h          :LOGDAC data port
001C          daccont equ 1ch          :LOGDAC control port

0080          intc11 equ 80h          :interrupt clear flip-flop's
0081          intc12 equ 81h

0082          trig equ 82h          :scope trigger port

          .z80
0000'         cseg

*****
;*
;*   Subroutine : Test
;*   Function : Dip switch test procedure
;*
;*   select
;*
;*   Inputs : A = 1..15 = test proc. number
;*   Outputs : none
;*
*****

0000'         test:
0000' 3D          dec a          :jump to function number in A
0001' 87          add a,a
0002' 5F          ld e,a
0003' 16 00        ld d,0
0005' 21 000E'     ld hl,jmptab
0008' 19          add hl,de

```

MACRO-80 3.44 09-Dec-81 PAGE 1-1

```

0009' 7E          ld      a,(h1)
000A' 23          inc     h1
000B' 66          ld      h,(h1)
000C' 6F          ld      l,a
000D' E9          jp      (h1)

000E'             jmpTAB:           ;function address table
000E' 002E'       dw      proctest  ;func. 1 = processor test
0010' 0044'       dw      rs_test   ;func. 2 = RS-232 test
0012' 004B'       dw      loopback  ;func. 3 = RS-232 loop-back test
0014' 005C'       dw      pal_test   ;func. 4 = PAL test
0016' 0083'       dw      del_test   ;func. 5 = delay line test
0018' 009D'       dw      lburst     ;func. 6 = Loran burst generator test
001A' 00DD'       dw      single     ;func. 7 = single burst output
001C' 00E9'       dw      burst8     ;func. 8 = 8 burst generation
001E' 00F5'       dw      ramp       ;func. 9 = ramp generation
0020' 0101'       dw      gnd_adj     ;func.10 = gndwave LOGDAC adjust program
0022' 013C'       dw      sky_adj     ;func.11 = skywave LOGDAC adjust program
0024' 0177'       dw      gri_test    ;func.12 = GRI generation
0026' 0183'       dw      ngri_test   ;func.13 = GRI generation with noise
0028' 002C'       dw      abort       ;func.14 = illegal number, abort
002A' 002C'       dw      abort       ;func.15 = illegal number, abort

002C'             abort:           ;stop processing
002C' 18 FE       jr      abort      ;endless loop...

```

```

;*****
;*
;*      Subroutine :   Proctest
;*      Function :    processor testing, generate
;*                   1 kHz square wave at trigger
;*                   output
;*      Inputs :      none
;*      Outputs :     none
;*
;*****

```

```

002E'             proctest:
002E' 3E 00       ld      a,0         ;trig --> 0
0030' 03 82       out     (trig),a
0032' 21 0005     ld      hl,5
0035' CD 01B9'    call    delay       ;delay 500 us
0038' 3E 01       ld      a,1
003A' 03 82       out     (trig),a    ;trig --> 1
003C' 21 0005     ld      hl,5
003F' CD 01B9'    call    delay       ;delay 500 us
0042' 1B EA       jr      proctest

```

```

;*****
;*
;*      Subroutine :   RS_test
;*      Function :     test RS-232interface
;*      Inputs :      none
;*      Outputs :     none
;*
;*****

```


MACRO-80 3.44 09-Dec-81 PAGE 1-2

```

0044'          rs_test:
0044' 3E 55          ld    a,'U'
0046' CD 0000*      call  send##
0049' 18 F9          jr    rs_test

;*****
;*
;* Subroutine :    Loopback
;* Function :      RS-232 Loop-back test, echo
;*                 received characters
;* Inputs :        data in input buffer
;* Outputs :       none
;*
;*****

004B'          loopback:
004B' 3E 00          ld    a,0          ;disable timer interrupts
004D' D3 80          out   (intc11),a
004F' D3 81          out   (intc12),a

0051' ED 56          im    1          ;enable RXRDY interrupts
0053' FB            ei

0054'          loopb1:
0054' CD 0000*      call  get_byt##      ;get byte from buffer
0057' CD 0000*      call  send##        ;& send it
005A' 18 F8          jr    loopb1

;*****
;*
;* Subroutine :    Pal_test
;* Function :      PAL testing, toggle PAL's
;*                 between various delay times
;* Inputs :        none
;* Outputs :       none
;*
;*****

005C'          pal_test:
005C' CD 0194*      call  set_tim        ;init timers

005F'          loop1:
005F' 06 00          ld    b,0          ;PAL delay = 0
0061' CD 007B*      call  setdelay
0064' 21 2710       ld    hl,10000      ;wait 1 sec
0067' CD 0189*      call  delay

006A'          loop2:
006A' 04            inc    b            ;variable PAL delay
006B' CD 007B*      call  setdelay
006E' 21 01F4       ld    hl,500
0071' CD 0189*      call  delay          ;wait 50 msec
0074' 78            ld    a,b
0075' FE 31         cp    49
0077' 20 F1         jr    nz,loop2      ;repeat

```

MACRO-80 3.44 09-Dec-81 PAGE 1-3

```

0079' 18 E4          jr      loop1

007B'               setdelay:          ;set PAL's to delay in B
007B' 3E 31          ld      a,49
007D' 90             sub      b
007E' D3 04          out     (grdde),a
0080' D3 08          out     (skyde),a
0082' C9             ret

*****
*                               *
* Subroutine : Del_test         *
* Function : delay line test, toggle delay *
*          : lines from 0...175 ns *
* Inputs : none *
* Outputs : none *
*                               *
*****

0083'               del_test:
0083' CD 0194'       call    set_tim     ;init timers
0086'               loop3:
0086' 06 00          ld      b,0
0088'               loop4:
0088' 78             ld      a,b          ;set bits for both delay lines
0089' 87             add     a,a
008A' 87             add     a,a
008B' 87             add     a,a
008C' 80             or      b
008D' D3 0C          out     (delport),a ;set delay line
008F' 21 0002        ld      hl,2
0092' CD 01B9'       call    delay       ;delay 200 us

0095' 04             inc     b
0096' 7B             ld      a,b          ;keep count
0097' FE 08          cp      8
0099' 20 ED          jr      nz,loop4

009B' 18 E9          jr      loop3       ;repeat

*****
*                               *
* Subroutine : Lburst          *
* Function : Loran burst generator test, *
*          : initialize hardware to generate *
*          : Loran bursts with 1 ms intervals*
* Inputs : none *
* Outputs : none *
*                               *
*****

009D'               lburst:
009D' 3E 00          ld      a,0          ;set ECD
009F' D3 10          out     (gndec),a
00A1' D3 14          out     (skyecd),a
00A3' CD 00A9'       call    make_burst

```

MACRO-B0 3.44 09-Dec-81 PAGE 1-4

```

00A6' C3 002C'          jp      abort

;*****
;*
;* Subroutine :      make_burst
;* Function :      initialize timers & LOGDAC's
;*               for burst generation
;* Inputs :      none
;* Outputs :      none
;*
;*****

00A9'          make_burst:
00A9' CD 0194'          call set_tim      :init timers

00AC' 3E 64          ld      a,100      :timer 1 = 1 ms interval
00AE' D3 01          out     (timer1),a
00B0' 3E 00          ld      a,0
00B2' D3 01          out     (timer1),a

00B4' 3E 28          ld      a,40      :skywave delay = 400 us
00B6' D3 02          out     (timer2),a
00B8' 3E 00          ld      a,0
00BA' D3 02          out     (timer2),a

00BC' 3E 00          ld      a,0      :init PAL's
00BE' D3 04          out     (gnddel),a
00C0' D3 08          out     (skydel),a
00C2' D3 0C          out     (delport),a :init delay lines

00C4' 3E 00          ld      a,0      :max. gndwave level
00C6' D3 18          out     (dacdata),a
00C8' 3E 06          ld      a,00000110b
00CA' D3 1C          out     (daccont),a
00CC' CB C7          set     0,a
00CE' D3 1C          out     (daccont),a

00D0' 3E FF          ld      a,255      :disable skywave & noise DAC's
00D2' D3 18          out     (dacdata),a
00D4' 3E 01          ld      a,00000001b
00D6' D3 1C          out     (daccont),a
00D8' 3E 1F          ld      a,00011111b
00DA' D3 1C          out     (daccont),a

00DC' C9          ret

;*****
;*
;* Subroutine :      Single
;* Function :      Single burst generator test.
;*               initialize hardware to generate
;*               single bursts with 1 ms
;*               intervals
;* Inputs :      none
;* Outputs :      none
;*
;*****

```


MACRO-80 3.44 09-Dec-81 PAGE 1-5

```

*****
00DD'      Single:
00DD'      3E 15      1d      a,21      ;select burst type
00DF'      D3 10      out      (gndecd),a
00E1'      D3 14      out      (skyeecd),a
00E3'      CD 00A9'    call     make_burst
00E6'      C3 002C'    jp       abort

```

```

*****
;*
;*      Subroutine :      Burst8
;*      Function :      Burst generator test.
;*      initialize hardware to generate
;*      8 bursts with 1 ms intervals
;*      Inputs :      none
;*      Outputs :      none
;*
*****

```

```

00E9'      burst8:
00E9'      3E 16      1d      a,22      ;select burst type
00EB'      D3 10      out      (gndecd),a
00ED'      D3 14      out      (skyeecd),a
00EF'      CD 00A9'    call     make_burst
00F2'      C3 002C'    jp       abort

```

```

*****
;*
;*      Subroutine :      Ramp
;*      Function :      Ramp generator test.
;*      initialize hardware to generate
;*      ramp bursts with 1 ms intervals
;*      Inputs :      none
;*      Outputs :      none
;*
*****

```

```

00F5'      ramp:
00F5'      3E 17      1d      a,23      ;select burst type
00F7'      D3 10      out      (gndecd),a
00F9'      D3 14      out      (skyeecd),a
00FB'      CD 00A9'    call     make_burst
00FE'      C3 002C'    jp       abort

```

```

*****
;*
;*      Subroutine :      Gnd_adj
;*      Function :      Gndwave LOGDAC feed-through
;*      adjust program, toggles gndwave
;*      burst on/off
;*      Inputs :      none
;*      Outputs :      none
;*
*****

```


MACRO-80 3.44 09-Dec-81 PAGE 1-6

```

0101'          gnd_adj:
0101' 3E 00      ld    a,0          ;select ECD
0103' D3 10      out   (gndecd),a
0105' D3 14      out   (skyeecd),a

0107' CD 00A9'   call  make_burst   ;start burst generators

010A' 3E FF      ld    a,255       ;disable LOGDAC's
010C' D3 18      out   (dacdata),a
010E' 3E 00      ld    a,00000000b
0110' D3 1C      out   (daccont),a
0112' 3E 1F      ld    a,00011111b
0114' D3 1C      out   (daccont),a
0116'          g_adjlp:
0116' 3E 00      ld    a,0          ;enable gndwave
0118' D3 18      out   (dacdata),a
011A' 3E 16      ld    a,00010110b
011C' D3 1C      out   (daccont),a
011E' 3E 1F      ld    a,00011111b
0120' D3 1C      out   (daccont),a

0122' 21 1388    ld    hl,5000     ;delay 0.5 sec
0125' CD 01B9'   call  delay

0128' 3E FF      ld    a,255       ;disable gndwave
012A' D3 18      out   (dacdata),a
012C' 3E 16      ld    a,00010110b
012E' D3 1C      out   (daccont),a
0130' 3E 1F      ld    a,00011111b
0132' D3 1C      out   (daccont),a

0134' 21 1388    ld    hl,5000     ;delay 0.5 sec
0137' CD 01B9'   call  delay

013A' 18 DA      jr     g_adjlp

```

```

*****
;*                                     *
;* Subroutine :   Sky_adj              *
;* Function :    Skywave LOGDAC feed-through *
;*               adjust program, toggles Skywave *
;*               burst on/off           *
;*                                     *
;* Inputs :      none                  *
;* Outputs :     none                  *
;*                                     *
*****

```

```

013C'          sky_adj:
013C' 3E 00      ld    a,0          ;select ECD
013E' D3 10      out   (gndecd),a
0140' D3 14      out   (skyeecd),a

0142' CD 00A9'   call  make_burst   ;start burst generators

0145' 3E FF      ld    a,255       ;disable LOGDAC's
0147' D3 18      out   (dacdata),a

```

MACRO-80 3.44 09-Dec-81 PAGE 1-7

```

0149' 3E 00          ld    a,00000000b
014B' 03 1C          out    (daccont),a
014D' 3E 1F          ld    a,00011111b
014F' 03 1C          out    (daccont),a
0151'               s_adjlp:
0151' 3E 00          ld    a,0           ;enable skywave
0153' 03 18          out    (dacdata),a
0155' 3E 25          ld    a,00100101b
0157' 03 1C          out    (daccont),a
0159' 3E 2F          ld    a,00101111b
015B' 03 1C          out    (daccont),a

015D' 21 1388        ld    hl,5000       ;delay 0.5 sec
0160' CD 01B9'       call    delay

0163' 3E FF          ld    a,255        ;disable skywave
0165' 03 18          out    (dacdata),a
0167' 3E 25          ld    a,00100101b
0169' 03 1C          out    (daccont),a
016B' 3E 2F          ld    a,00101111b
016D' 03 1C          out    (daccont),a

016F' 21 1388        ld    hl,5000       ;delay 0.5 sec
0172' CD 01B9'       call    delay

0175' 18 DA          jr     s_adjlp

;*****
;*
;* Subroutine :    gri_test
;* Function :      Generate test GRI. 1 master,
;*                 2 slaves with different levels
;* Inputs :        none
;* Outputs :       none
;*
;*****

0177'               gri_test:
0177' CD 0000*       call    initbrs##    ;init burst processing & set default parameters

017A' ED 56          im     1           ;interrupt mode 1
017C' FB             ei              ;enable burst interrupts

017D' CD 0000*       call    start##     ;start burst generator
0180' C3 002C'       jp      abort      ;stop processor

;*****
;*
;* Subroutine :    ngri_test
;* Function :      Generate test GRI. 1 master,
;*                 2 slaves with different levels
;*                 with noise
;* Inputs :        none
;* Outputs :       none
;*
;*****

```


MACRO-80 3.44 09-Dec-81 PAGE 1-8

```

0183'          ngri_test:
0183'  CD 0000*          call  initbrs##          ;init burst processing & set default parameters

0186'  3E 00          ld    a,0              ;max. noise level
0188'  32 0000*       ld    (noise##),a

0188'  ED 56          im    1              ;interrupt mode 1
018D'  FB            ei                    ;enable burst interrupts

018E'  CD 0000*       call  start##          ;start burst generator
0191'  C3 002C'       jp    abort          ;stop processor

```

```

*****
;*
;*      Subroutine :   Set_tim              *
;*      Function :    Initilize timers for testing *
;*      Inputs :      none                  *
;*      Outputs :     none                  *
;*
*****

```

```

0194'          set_tim:
0194'  3E 34          ld    a,00110100b      ;init timer 0
0196'  D3 03          out   (timmod),a      ;mode 2 (rate generator)
0198'  3E 32          ld    a,50           ;generate 100 kHz pulse
019A'  D3 00          out   (timer0),a
019C'  3E 00          ld    a,0
019E'  D3 00          out   (timer0),a

01A0'  3E 74          ld    a,01110100b    ;init timer 1
01A2'  D3 03          out   (timmod),a      ;mode 2 (rate generator)

01A4'  3E 0A          ld    a,10           ;dummy load for testing
01A6'  D3 01          out   (timer1),a
01A8'  3E 00          ld    a,0
01AA'  D3 01          out   (timer1),a

01AC'  3E BA          ld    a,10111010b    ;init timer 2
01AE'  D3 03          out   (timmod),a      ;mode 5 (hardware triggered strobe)

01B0'  3E 02          ld    a,2           ;dummy load for testing
01B2'  D3 02          out   (timer2),a
01B4'  3E 00          ld    a,0
01B6'  D3 02          out   (timer2),a

01B8'  C9            ret

```

```

*****
;*
;*      Subroutine :   Delay              *
;*      Function :     software delay routine *
;*      Inputs :      HL = delay in 100 us units *
;*      Outputs :     none                  *
;*
*****

```

MACRO-80 3.44 09-Dec-81 PAGE 1-9

```
01B9'          delay:
01B9' 3E 22          ld    a,34
01BB'          dell:
01BB' 3D          dec    a
01BC' C2 01BB'      jp     nz,dell
01BF' 2B          dec    hl
01C0' 7C          ld     a,h
01C1' 85          or     l
01C2' C2 01B9'      jp     nz,delay
01C5' C9          ret

          end
```

MACRO-80 3.44 09-Dec-81 PAGE 5

Macros:

Symbols:

002C*	ABORT	00E9*	BURST8	001C	DACCONT
0018	DACDATA	0188*	DEL1	0189*	DELAY
000C	DELPQRT	0083*	DEL_TEST	0055*	GET_BYT
0004	GNDDEL	0010	GNDPCD	0101*	GND_ADJ
0177*	GRI_TEST	0116*	G_ADJLP	0184*	INITBRS
0080	INTCL1	0081	INTCL2	000E*	JMPTAB
009D*	LBURST	005F*	LOOP1	006A*	LOOP2
00B6*	LOOP3	00BB*	LOOP4	0054*	LOOPB1
004B*	LOOPBACK	00A9*	MAKE_BURST	0183*	NGRI_TEST
0189*	NOISE	005C*	PAL_TEST	002E*	PROCTEST
00F5*	RAMP	0044*	RS_TEST	0058*	SEND
007B*	SETDELAY	0194*	SET_TIM	000D*	SINGLE
0008	SKYDEL	0014	SKYPCD	013C*	SKY_ADJ
018F*	START	0151*	S_ADJLP	0000I*	TEST
0000	TIMER0	0001	TIMER1	0002	TIMER2
0003	TIMMOD	00B2	TRIG		

No Fatal error(s)

□

C. Source code listings of modified software

MACRO-80 3.44 09-Dec-81

PAGE 1

```

/*****
*
*      SIM.MAC      Eurofix Simulator Main Program
*
*
*      Version      : 0
*      Last modification : 15/02/1993
*      Author       : A.M. Noorbergen
*
*      Modified version of SIM.MAC written by R. Kellenbach
*      (C) Delft University of Technology, 1987, 1993
*
*      17/02/1993 : Eurofix mode can be set for each station separately
*                  ESC T command added. ESC R and ESC S have become
*                  obsolete and were deleted
*
*****/

```

```

001B      ESC      equ      1bh      ;escape code
001D      GS       equ      1dh      ;rel. move command code

0080      intc11    equ      80h      ;interrupt clear flip-flop's

0091      uartc     equ      91h      ;8251 status register
00A0      swport    equ      0a0h     ;dip switch input port

00B6      noisep    equ      86h      ;noise control port

```

```

      .z80
0000*     cseg

      .request simio,simbrs,simtest

```

```

0000* 31 0080*      ld      sp,stack      ;init stack

0003* CD 0000*      call    initrs##      ;init RS-232 interface

0006*      boot:      ;restart address

0006* CD 0000*      call    initbrs##     ;init burst processing

0009* 3E 01      ld      a,1      ;start noise generator
000B* D3 86      out      (noisep),a
000D* 3E 00      ld      a,0
000F* D3 86      out      (noisep),a

0011* DB A0      in       a,(swport)    ;get switch status
0013* 2F      cpl
0014* E6 0F      and      0fh      ;switch 1..4
0016* C2 0000*    jp       nz,test##    ;hardware testing ?

0019* ED 56      im       1      ;interrupt mode 1

```

MACRO-80 3.44 09-Dec-81 PAGE 1-1

```

001B' FB          ei          :enable interrupts

001C' C3 0053'    jp    get_cmd    :skip interrupt vector

;*****
;*
;*      Subroutine :    Interrupt
;*      Function :    Uart & Timer interrupt
;*                  processing
;*      Inputs :    interrupt flags in Uart status
;*                  register
;*      Outputs :    none
;*
;*****

          org    38h          :mode 1 interrupt vector

0038' F5          push    af          :save registers
0039' C5          push    bc
003A' D5          push    de
003B' E5          push    hl

003C' Z1 004D'    ld    hl,exit_int    :push return address on stack
003F' E5          push    hl

0040' DB 91          in    a,(uartc)    :get interrupt flag bits

0042' CB 7F          bit    7,a          :burst timer interrupt ?
0044' CA 0000*      jp    z,brs_int##

0047' CB 4F          bit    1,a          :Uart RXRDY interrupt ?
0049' C2 0000*      jp    nz,rx_int##

004C' C9          ret          :other interrupts not allowed

004D'          exit_int:          :exit interrupt routine
004D' E1          pop     hl          :restore registers
004E' D1          pop     de
004F' C1          pop     bc
0050' F1          pop     af
0051' FB          ei          :enable interrupts again
0052' C9          ret

```

```

;*****
;*
;*      Subroutine :    Get_cmd
;*      Function :    Main program command processing
;*      Inputs :    commands in input data buffer
;*      Outputs :    burst processing data tables
;*                  updated
;*
;*****

```


MACRO-80 3.44 09-Dec-81 PAGE 1-2

```

0053'          get_cmd:

0053' 21 0053'      ld    hl,get_cmd    ;push return address on stack
0056' E5          push  hl

0057' CD 0000*     call  get_byt##      ;get command character

005A' FE 1D       cp     GS            ;rel. move command ?
005C' CA 00FD'    jp     z,rel_mov

005F' FE 1B       cp     ESC          ;ESCAPE command ?
0061' CA 0065'    jp     z,esc_cmd

0064' C9         ret                  ;invalid command, ignore

0065'          esc_cmd:                ;escape command processing
0065' CD 0000*     call  get_byt##      ;get command character
0068' FE 41       cp     'A'          ;command A-Z ?
006A' D8         ret     c
006B' FE 5B       cp     'Z'+1
006D' D0         ret     nc

006E' D6 41       sub     'A'          ;perform table look-up
0070' 87         add     a,a
0071' 5F         ld     e,a
0072' 16 00       ld     d,0
0074' 21 007D'    ld     hl,cmd_tab
0077' 19         add     hl,de
0078' 7E         ld     a,(hl)
0079' 23         inc     hl
007A' 66         ld     h,(hl)
007B' 6F         ld     l,a
007C' E9         jp     (hl)          ;jump to selected function

007D'          cmd_tab:                ;escape command address table
007D' 0120'       dw     no_stat      ;ESC A - set # stations
007F' 0128'       dw     set_dta     ;ESC B - set initial dta's
0081' 00B1'       dw     invalid     ;ESC C - not used
0083' 0155'       dw     set_sdel    ;ESC D - set skywave delays
0085' 0183'       dw     set_glev    ;ESC E - set gndwave att. level
0087' 0196'       dw     set_slev    ;ESC F - set skywave att. level
0089' 01A9'       dw     set_gecd    ;ESC G - set gndwave ECD
008B' 01BC'       dw     set_secd    ;ESC H - set skywave ECD
008D' 01CF'       dw     set_sign    ;ESC I - set sign code
008F' 01D9'       dw     set_noise   ;ESC J - set noise level
0091' 01E1'       dw     reset       ;ESC K - reset simulator
0093' 01E9'       dw     set_0patt   ;ESC L - set zero pattern (Eurofix mode only)
0095' 01F9'       dw     set_0shft   ;ESC M - set zero shifts (Eurofix mode only)
0097' 01F1'       dw     set_1patt   ;ESC N - set one pattern (Eurofix mode only)
0099' 0201'       dw     set_1shft   ;ESC O - set one shifts (Eurofix mode only)
009B' 0209'       dw     set_advnc   ;ESC P - set phase advance (Eurofix mode only)
009D' 0211'       dw     set_delay   ;ESC Q - set phase delay (Eurofix mode only)
009F' 00B1'       dw     invalid     ;ESC R - not used
00A1' 00B1'       dw     invalid     ;ESC S - not used

```

MACRO-80 3.44 09-Dec-81 PAGE 1-3

```

00A3' 0219'      dw      set_eurof      ;ESC T = set Eurofix mode
00A5' 00B1'      dw      invalid      ;ESC U = not used
00A7' 00B1'      dw      invalid      ;ESC V = not used
00A9' 00B1'      dw      invalid      ;ESC W = not used
00AB' 00B1'      dw      invalid      ;ESC X = not used
00AD' 00B1'      dw      invalid      ;ESC Y = not used
00AF' 00B1'      dw      invalid      ;ESC Z = not used

```

```

00B1'      invalid:      ;invalid command, return
00B1' C9      ret

```

```

;*****
;*
;*      Subroutine :   Get_int
;*
;*      Function :   get integer parameter,
;*                  (signed or unsigned)
;*
;*                  max 32 bits.
;*
;*      Inputs :   none
;*
;*      Outputs :   DEHL = 32 bit parameter
;*
;*****

```

```

00B2'      get_int:
00B2' C5      push     bc      ;save BC
00B3' 21 0000  ld      hl,0      ;DEHL = 0
00B6' 11 0000  ld      de,0
00B9'      getl:
00B9' CD 0000* call     get_byt      ;get byte
00BC' FE 20      cp      20h
00BE' 38 F9      jr      c.getl      ;ignore control characters
00C0' FE 40      cp      40h
00C2' 38 0A      jr      c.int_lo      ;low or high byte ?
00C4' E6 3F      and     00111111b      ;high byte : only 6 bits
00C6' 4F      ld      c,a
00C7' 06 06      ld      b,b      ;multiply result by 64 and add new bits
00C9' CD 00F0' call     shift
00CC' 18 EB      jr      getl
00CE'      int_lo:
00CE' F5      push     af      ;low byte
00CF' E6 0F      and     00001111b      ;only 4 bits
00D1' 4F      ld      c,a
00D2' 06 04      ld      b,b      ;multiply result by 16 and add new bits
00D4' CD 00F0' call     shift
00D7' F1      pop      af      ;get sign-bit
00D8' E6 10      and     10h
00DA' 20 12      jr      nz,int_rdy      ;sign-bit set ?

00DC' 7D      ld      a,l      ;invert : DEHL = -DEHL
00DD' 2F      cpl
00DE' 6F      ld      l,a
00DF' 7C      ld      a,h
00E0' 2F      cpl
00E1' 67      ld      h,a
00E2' 7B      ld      a,e
00E3' 2F      cpl
00E4' 5F      ld      e,a

```


MACRO-80 3.44 09-Dec-81 PAGE 1-4.

```

00E5' 7A          ld    a,d
00E6' 2F          cpl
00E7' 57          ld    d,a
00E8' 23          inc   hl
00E9' 7C          ld    a,h
00EA' 85          or     l
00EB' 20 01       jr     nz,int_rdy
00ED' 13          inc   de
00EE'             int_rdy:
00EE' C1          pop    bc
00EF' C9          ret

00F0'             shift:                ;rotate DEHL 8 times and add C
00F0' 29          add    hl,hl
00F1' EB          ex     de,hl          ;DEHL = DEHL * 2
00F2' ED 6A       adc    hl,hl
00F4' EB          ex     de,hl
00F5' 10 F9       djnz   shift
00F7' 06 00       ld     b,0            ;add new bits in C
00F9' 09          add    hl,bc
00FA' 00          ret     nc
00FB' 13          inc   de
00FC' C9          ret

;*****
;*
;* Subroutine : Rel_mov
;* Function : Relative Id move for all
;* stations
;* Inputs : <int>-parameters in input buffer*
;*          -128...+127 * 25 ns steps
;* Outputs : Mov_flg's set
;*           Mov_tim updated
;*
;*****

00FD'             rel_mov:
00FD' 3A 0000*     ld     a,(hstat##)    ;B = station count
0100' 47          ld     b,a

0101' DD 21 0000* ld     ix,mov_flg##    ;IX = flag pointer
0105' FD 21 0000* ld     iy,mov_tim##    ;IY = pointer to shift time table
0109'             hold:
0109' DD CB 00 46 bit    0,(ix+0)        ;hold if prev. move request not processed yet
010D' 20 FA       jr     nz,hold

010F' CD 00B2'     call   get_int        ;get shift time
0112' FD 75 00     ld     (iy+0),l      ;& store it

0115' DD CB 00 C6 set    0,(ix+0)        ;set move request flag

0119' DD 23        inc    ix            ;move pointers
011B' FD 23        inc    iy

011D' 10 EA       djnz   hold          ;keep count

```


MACRO-80 3.44 09-Dec-81 PAGE 1-5

011F* C9 ret

```

*****
;*
;* Subroutine : No_stat
;* Function : Set # stations in GRI
;* Inputs : <int> parameter in buffer
;*           = # stations
;* Outputs : Nstat = # stations
;*
*****

```

0120* no_stat:

0120* CD 00B2* call get_int :get parameter

0123* 7D ld a,l

0124* 32 0000* ld (nstat##),a ;& store it

0127* C9 ret

```

*****
;*
;* Subroutine : Set_DTA
;* Function : Set initial dta's
;* Inputs : <int> parameters in buffer,
;*           24 bit in 25 ns units
;* Outputs : dtatab updated
;*
*****

```

0128* set_dta:

0128* 3A 0000* ld a,(nstat##) ; B = station count

0129* 47 ld b,a

012C* 00 21 0000* ld ix,dtatab## ; IX = DTA destination pointer

0130* dta_lp:

0130* CD 00B2* call get_int ; DTA in EHL

0133* 16 00 ld d,0

0135* EB ex de,hl ; HLDE now contains DTA

0136* C5 push bc

0137* 01 0190 ld bc,400 ; divide HLDE by 400 to

013A* CD 0000* call div32## ; split into 10us and 25ns units

013D* C1 pop bc

013E* 00 73 00 ld (ix+0),e ;store DTA DE = quotient

0141* 00 72 01 ld (ix+1),d ; HL = remainder

0144* 00 75 02 ld (ix+2),l

0147* 00 74 03 ld (ix+3),h

014A* 00 23 inc ix ;move pointer to next delay

014C* 00 23 inc ix

014E* 00 23 inc ix

0150* 00 23 inc ix

0152* 10 DC djnz dta_lp ;keep count

```

MACRO-80 3.44    09-Dec-81    PAGE    1-6

0154'  C9                      ret

;*****
;*
;*      Subroutine :    Set_sdel
;*      Function :    Set skywave delays
;*      Inputs :    <int> parameters in buffer.
;*                  16 bit delays in 25 ns units
;*      Outputs :    Skytab updated
;*
;*****

0155'          set_sdel:
0155'  3A 0000*          ld    a,(nstat##)    ;B = station count
0158'  47                ld    b,a

0159'  DD 21 0000*          ld    ix,skytab##    ;IX = destination pointer

0150'          sdel_lp:
0150'  CD 00B2'          call    get_int        ;HL = skywave delay

0160'  EB                ex    de,hl          ;move it to DE
0161'  21 0000          ld    hl,0
0164'  C5                push    bc
0165'  01 0190          ld    bc,400          ;split delay into 10us part and 25ns remainder
0168'  CD 0000*          call    div32##

016B'  DD 73 00          ld    (ix+0),e        ;store skywave delay    DE = quotient
016E'  DD 72 01          ld    (ix+1),d        ;                      HL = remainder
0171'  DD 75 02          ld    (ix+2),l
0174'  DD 74 03          ld    (ix+3),h

0177'  DD 23                inc    ix          ;move pointer to next delay
0179'  DD 23                inc    ix
017B'  DD 23                inc    ix
017D'  DD 23                inc    ix

017F'  C1                pop    bc
0180'  10 DB            djnz    sdel_lp

0182'  C9                      ret

;*****
;*
;*      Subroutine :    Set_glev
;*      Function :    Set gndwave att. level
;*      Inputs :    <int> parameters in data buffer
;*                  8 bit attenuation in dB's
;*      Outputs :    Gnd_lev updated
;*
;*****

0183'          set_glev:
0183'  3A 0000*          ld    a,(nstat##)    ;B = station count

```


MACRO-80 3.44 09-Dec-81 PAGE 1-7

```

0186' 47          ld      b,a

0187' DD 21 0000*          ld      ix,gnd_lev##      ;IX = destination pointer
0188'          glev_lp:
018B' CD 00B2'          call   get_int      ;L = att. level
018E' DD 75 00          ld      (ix+0),l      ;store it

0191' DD 23          inc     ix              ;move pointer

0193' 10 F6          djnz   glev_lp

0195' C9          ret

```

```

*****
;*
;*      Subroutine :   Set_slev
;*      Function :    Set skywave att. level
;*      Inputs :      <int> parameters in data buffer
;*                   8 bit attenuation in dB's
;*      Outputs :     sky_lev updated
;*
*****

```

```

0196'          set_slev:
0196' 3A 0000*          ld      a,(nstat##)      ;B = station count
0199' 47          ld      b,a

019A' DD 21 0000*          ld      ix,sky_lev##      ;IX = destination pointer
019E'          slev_lp:
019E' CD 00B2'          call   get_int      ;L = att. level
01A1' DD 75 00          ld      (ix+0),l      ;store it

01A4' DD 23          inc     ix              ;move pointer

01A6' 10 F6          djnz   slev_lp

01A8' C9          ret

```

```

*****
;*
;*      Subroutine :   Set_gecd
;*      Function :     Set gndwave ECD's
;*      Inputs :      <int> parameters in data buffer
;*      Outputs :     g_ecdtb updated
;*
*****

```

```

01A9'          set_gecd:
01A9' 3A 0000*          ld      a,(nstat##)      ;B = station count
01AC' 47          ld      b,a

01AD' DD 21 0000*          ld      ix,g_ecdtb##      ;IX = destination pointer
01B1'          set_glp:
01B1' CD 00B2'          call   get_int      ;L = ECD code
01B4' DD 75 00          ld      (ix+0),l      ;store it

```

MACRO-80 3.44 09-Dec-81 PAGE 1-8

```
01B7' DD 23          inc    ix          ;move pointer
```

```
01B9' 10 F6          djnz   set_glp.
```

```
01BB' C9             ret
```

```
*****
;*
;* Subroutine : Set_secd
;* Function : Set skywave ECD's
;* Inputs : <int> parameters in data buffer
;* Outputs : s_ecdtb updated
;*
*****
```

```
01BC* set_secd:
01BC' 3A 0000*      ld      a,(nstat##) ;B = station count
01BF' 47             ld      b,a

01C0* DD 21 0000*      ld      ix,s_ecdtb## ;IX = destination pointer
```

```
01C4' set_slp:
01C4' CD 00B2*      call   get_int    ;L = ECD code
01C7' DD 75 00      ld      (ix+0),l ;store it
```

```
01CA' DD 23          inc    ix          ;move pointer
```

```
01CC' 10 F6          djnz   set_slp.
```

```
01CE' C9             ret
```

```
*****
;*
;* Subroutine : Set_sign
;* Function : Set output polarity
;* Inputs : <int> parameter in data buffer
;*          0 = positive polarity
;*          1 = negative polarity
;* Outputs : sign = sign code
;*
*****
```

```
01CF* set_sign:
01CF' CD 00B2*      call   get_int    ;get sign parameter
01D2' 7D             ld      a,l
01D3' E6 01          and     1          ;1 bit only
01D5' 32 0000*      ld      (sign##),a ;store it
01D8' C9             ret
```

```
*****
;*
;* Subroutine : Set_noise
;* Function : Set noise level
;* Inputs : <int> parameter in data buffer
;* Outputs : noise = noise level
;*
*****
```



```

*****
:*
:*
:*      Subroutine :      Reset
:*
:*      Function :      Resets simulator
:*
:*      Inputs :      none
:*
:*      Outputs :      none
:*
:*
:*
*****

```

```

*****
*
* Subroutine : Set_0patt
*
* Function : Set burst advance/delay pattern
*
* for 0-databit. Used for Eurofix
*
* mode only
*
* Inputs : <int> parameter in buffer,
*
* 8 bit
*
* Outputs : patt_0 updated
*
*
*****

```

```
*****  
.*  
.*      Subroutine :    Set_lptatt      *  
.*      Function  :    Set burst advance/delay pattern *  
.*                  for 1-databit. Used for Eurofix *  
.*
```


MACRO-80 3.44 09-Dec-81 PAGE 1-10

```

;*          mode only          *
;*   Inputs :   <int> parameters in buffer.  *
;*          8 bit              *
;*   Outputs :   patt_1 updated  *
;*          *                    *
;*****
01F1'          set_1patt:

01F1'  CD 00B2'          call  get_int      ;get parameter
01F4'  7D              ld      a,1
01F5'  32 0000*        ld      (patt_1##),a

01F8'  C9              ret

;*****
;*          *
;*   Subroutine :   Set_0shft          *
;*   Function :     Set burst shifts (pos/neg)  *
;*                for 0-databit. Used for Eurofix *
;*                mode only              *
;*   Inputs :       <int> parameters in buffer.  *
;*                8 bit                  *
;*   Outputs :      shift_0 updated          *
;*                *                      *
;*****
01F9'          set_0shft:

01F9'  CD 00B2'          call  get_int      ;get parameter
01FC'  7D              ld      a,1
01FD'  32 0000*        ld      (shift_0##), a

0200'  C9              ret

;*****
;*          *
;*   Subroutine :   Set_1shft          *
;*   Function :     Set burst shifts (pos/neg)  *
;*                for 1-databit. Used for Eurofix *
;*                mode only              *
;*   Inputs :       <int> parameters in buffer.  *
;*                8 bit                  *
;*   Outputs :      shift_1 updated          *
;*                *                      *
;*****
0201'          set_1shft:

0201'  CD 00B2'          call  get_int      ;get parameter
0204'  7D              ld      a,1
0205'  32 0000*        ld      (shift_1##),a

```

MACRO-80 3.44 09-Dec-81 PAGE 1-11

0208' C9 ret

```

*****
;*
;* Subroutine : Set_advnc
;* Function : Set Eurofix phase advance time
;* Used for Eurofix mode only
;* Inputs : <int> parameters in buffer
;* 8 bit in 25 ns units
;* Outputs : adv_tim updated
;*
*****

```

0209' set_advnc:

```

0209' CD 00B2' call get_int :get parameter
020C' 7D ld a,l
020D' 32 0000* ld (adv_tim##),a

```

0210' C9 ret

```

*****
;*
;* Subroutine : Set_delay
;* Function : Set Eurofix phase delay time
;* Used for Eurofix mode only
;* Inputs : <int> parameters in buffer
;* 8 bit in 25 ns units
;* Outputs : del_tim updated
;*
*****

```

0211' set_delay:

```

0211' CD 00B2' call get_int :get parameter
0214' 7D ld a,l
0215' 32 0000* ld (del_tim##),a

```

0218' C9 ret

```

*****
;*
;* Subroutine : Set_eurof
;* Function : Set eurofix mode for each station
;* Inputs : <int> parameters in data buffer
;* Outputs : mod_tab updated
;*
*****

```

MACRO-80 3.44 09-Dec-81 PAGE 1-12

```

0219'          set_eurof:
0219' 3A 0000*      ld    a,(nstat##)    ;B = station count
021C' 47          ld    b,a

021D' 00 21 0000*  ld    ix,mod_tab##    ;IX = destination pointer
0221'          set_elp:
0221' CD 0082'     call  get_int         ;L = eurofix mode
0224' 00 75 00     ld    (ix+0),l       ;store it
0227' 00 23       inc    ix             ;move pointer
0229' 10 F6       djnz  set_elp
022B' C9         ret

```

```

022C'          dseg          ;data area
0000"          ds    128      ;stack area
0080"          stack:
0080" 00          db    0
          end

```


MACRO-80 3.44 09-Dec-81 PAGE 5

Macros:

Symbols:

020E*	ADV_TIM	0006*	BOOT	0045*	BRS_INT
007D*	CMD_TAB	0216*	DEL_TIM	0169*	DIV32
012E*	DTATAB	0130*	DTA_LP	001B	ESC
0065*	ESC_CMD	004D*	EXIT_INT	00B9*	GET1
00BA*	GET_BYT	0053*	GET_CMD	00B2*	GET_INT
018B*	GLEV_LP	0189*	GND_LEV	001D	GS
01AF*	G_ECDB	0109*	HOLD	0007*	INITBRS
0004*	INITRS	0080	INTCL1	00CE*	INT_LO
00EE*	INT_RDY	00B1*	INVALID	021F*	MOD_TAB
0103*	MOV_FLG	0107*	MOV_TIM	01DE*	NOISE
0086	NOISEP	0120*	NO_STAT	021A*	NSTAT
01EE*	PATT_0	01F6*	PATT_1	00FD*	REL_MOV
01E1*	RESET	004A*	RX_INT	015D*	SDEL_LP
01E9*	SET_OPATT	01F9*	SET_0SHFT	01F1*	SET_1PATT
0201*	SET_1SHFT	0209*	SET_ADVNC	0211*	SET_DELAY
012B*	SET_DTA	0221*	SET_ELP	0219*	SET_EUROF
01A9*	SET_GECD	0183*	SET_GLEV	01B1*	SET_GLP
01D9*	SET_NOISE	0155*	SET_SDEL	01BC*	SET_SECD
01CF*	SET_SIGN	0196*	SET_SLEV	01C4*	SET_SLP
01FE*	SHFT_0	0206*	SHFT_1	00F0*	SHIFT
01D6*	SIGN	015B*	SKYTAB	019C*	SKY_LEV
019E*	SLEV_LP	0080*	STACK	00A0	SWPORT
01C2*	S_ECDB	0017*	TEST	0091	UARTC

No Fatal error(s)

□

MACRO-80 3.44 09-Dec-81 PAGE 1

```

;*****\
;*
;*      SIMIO.MAC          RS 232 I/O for Eurofix Simulator
;*
;*
;*
;*      Version           : 0
;*      Last modification  : 15/02/1993
;*      Author            : A.M. Noorbergen
;*
;*
;*      Modified version of SIMIO.MAC written by R. Kellenbach
;*      (C) Delft University of Technology, 1987, 1993
;*
;*****/

0019      strtcmd equ    19h          ;start command
001C      statcmd equ    1ch          ;buffer status request command
001A      datacmd equ    1ah          ;data command, data bits follow
001E      sel1  equ    1eh          ;simulator select codes
001F      sel2  equ    1fh

0090      uartd  equ    90h          ;8251 data register
0091      uartc  equ    91h          ;8251 control/status register

00A0      swport equ    0a0h          ;dip switch input port

1770      bsize equ    6000          ;host input buffer size
157C      bstop equ    5500          ;handshake thresholds
1388      bstart equ    5000

        .z80
        .request simbrs

0000'      cseg

cmphl macro val          ;16 bit compare macro
        local cmprdy
        ld    a,h          ;compare HL with <val>
        cp    high {val}
        jr    nz,cmprdy
        ld    a,l
        cp    low {val}
cmprdy:
        endm

comp16 macro              ; 16 bit compare macro
        ld    a,h          ; compare HL and DE
        cp    d
        ; zero flag set when equal
        jr    nz,co_rdy
        ld    a,l
        cp    e
co_rdy:
        endm

```


MACRO-80 3.44 09-Dec-81 PAGE 1-1

```

*****
;*
;*      subroutine :    initrs
;*      function :    initialize rs-232 interface
;*      inputs :    none
;*      outputs :    none
;*
*****

0000*      initrs::

                                ; i n i t   c o m m u n i c a t i o n   c o n t r o l l e r
0000*      3E CE                ld      a,11001110b      ;uart mode instruction
0002*      D3 91                out     (uartc),a        ;8 bits, no parity, 2 stop bits
0004*      3E 27                ld      a,00100111b      ;enable transmitter & receiver
0006*      D3 91                out     (uartc),a        ;DTR & RTS high
                                rept     3              ;dummy read's to clear uart
                                in       a,(uartd)
                                endm
0008*      DB 90                +      in       a,(uartd)
000A*      DB 90                +      in       a,(uartd)
000C*      DB 90                +      in       a,(uartd)

000E*      3E 00                ld      a,0              ;first byte to be expected surely is not
0010*      32 1778*            ld      (dataexp),a      ;not a databyte

                                ; i n i t   c o m m a n d   b u f f e r
0013*      21 0006*            ld      hl,buffer        ;init buffer pointers
0016*      22 0000*            ld      (rdpnt),hl
0019*      22 0002*            ld      (wrpnt),hl
001C*      21 0000            ld      hl,0              ;init byte count of command buffer
001F*      22 0004*            ld      (bufcnt),hl

                                ; i n i t   d a t a   b u f f e r
0022*      3E 00                ld      a, 0            ;init databuffer read pointer
0024*      32 0000*            ld      (datrd##), a
0027*      21 0000*            ld      hl,datwr##        ;init databuffer write pointers
002A*      06 07                ld      b,7
002C*      77                  14: ld      (hl), a
002D*      23                  inc     hl
002E*      05                  dec     b
002F*      C2 002C*            jp      nz, 14

                                ; d e t e r m i n e   s i m u l a t o r   a d d r e s s
0032*      DB A0                in       a,(swport)      ;read dip switch 8
0034*      CB 7F                bit      7,a
0036*      2B 04                jr      z,set_2          ;determine select code
0038*      3E 1E                ld      a,set1
003A*      1B 02                jr      set_ok
003C*      3E 1F                set_2: ld      a,set2

```

MACRO-80 3.44 09-Dec-81 PAGE 1-2

```

003E' 32 1776'      sel_ok: ld      (scode),a
0041' 3E 00          ld      a,0          ;clear select flag
0043' 32 1777'      ld      (selflg),a

0046' C9            ret

;*****
;*
;* Subroutine :    RX_int
;* Function :    Uart RX_RDY interrupt
;*               processing, read character &
;*               store in buffer
;* Inputs :    WRPNT = dest. pointer
;*             BUFcnt = byte count
;* Outputs :    WRPNT, BUFcnt incremented
;*             character in buffer
;*
;*****

0047'      rx_int::

; get character
0047' DB 90          in      a,(uartd)    ;get character & clear interrupt request
0049' 47            ld      b,a          ;save character

; process character
004A' 3A 1778'      ld      a,(dataexp)  ;are there databytes to be expected?
004D' FE 00          cp      0
004F' C2 00C9'      jp      nz,databytes

0052' 78            ld      a,b          ;get character

0053' FE 1A          cp      datacmd     ;data command ?
0055' CA 00C9'      jp      z,databytes

0058' FE 1E          cp      sel1        ;select code ?
005A' CA 00A9'      jp      z,select
005D' FE 1F          cp      sel2
005F' CA 00A9'      jp      z,select

0062' 3A 1777'      ld      a,(selflg)    ;simulator selected ?
0065' B7            or      a
0066' CB            ret      z          ;no, ignore character & return

0067' 78            ld      a,b          ;status request ?
0068' FE 1C          cp      statcmd
006A' CA 00BC'      jp      z,status

006D' FE 19          cp      strtcmd     ;start command ?
006F' CA 0000'      jp      z,start##

0072' 2A 0004'      ld      hl,(bufcnt)  ;get byte count
;cmphl bsize ;return if buffer full
0075' 7C            ld      a,h          ;compare HL with <val>

```


MACRO-80 3.44 09-Dec-81 PAGE 1-3

```

0076' FE 17      +      cp      high (bsize)
0078' 20 03      +      jr      nz,...0000
007A' 7D         +      ld      a,l
007B' FE 70      +      cp      low (bsize)
007D'           +      ..0000:
007D' C8         +      ret      z

007E' 23         +      inc     hl      ;increment buffer count
007F' 22 0004"   +      ld      (bufcnt),hl

                                cmphl   bstop      ;buffer almost full ?
0082' 7C         +      ld      a,h      ;compare HL with <val>
0083' FE 15      +      cp      high (bstop)
0085' 20 03      +      jr      nz,...0001
0087' 7D         +      ld      a,l
0088' FE 7C      +      cp      low (bstop)
008A'           +      ..0001:
008A' 20 04      +      jr      nz,rx_cont      ;no, continue

008C' 3E 25      +      ld      a,00100101b      ;handshake : DTR low
008E' D3 91      +      out     (uartc),a
0090'           +      rx_cont:
0090' 2A 0002"   +      ld      hl,(wrpnt)      ;get dest. pointer
0093' 78         +      ld      a,b
0094' E6 7F      +      and     7fh      ;only 7 bits
0096' 77         +      ld      (hl),a      ;store character
0097' 23         +      inc     hl      ;update pointer
                                cmphl   buffer+bsize      ;past end of buffer ?
0098' 7C         +      ld      a,h      ;compare HL with <val>
0099' FE 17"     +      cp      high (buffer+bsize)
009B' 20 03      +      jr      nz,...0002
009D' 7D         +      ld      a,l
009E' FE 76"     +      cp      low (buffer+bsize)
00A0'           +      ..0002:
00A0' 20 03      +      jr      nz,wr_ok
00A2' 21 0006"   +      ld      hl,buffer      ;wrap around
00A5'           +      wr_ok:
00A5' 22 0002"   +      ld      (wrpnt),hl      ;save pointer

00A8' C9         +      ret

;*****
;*
;*      Subroutine :      Select
;*      Function :      select/deselect simulator
;*      Inputs :      A = received select code
;*                   Scode = simulator select code
;*      Outputs :      Selflg = select flag
;*
;*****

00A9'           +      select:
00A9' 47         +      ld      b,a      ;compare A with simulator select code
00AA' 3A 1776"   +      ld      a,(scode)

```

MACRO-80 3.44 09-Dec-81 PAGE 1-4

```

00AD' B8          cp      b
00AE' 20 06       jr      nz,de_sel      :deselect simulator if not equal

00B0' 3E 01       ld      a,l          :select simulator
00B2' 32 1777*    ld      (selflg),a

00B5' C9          ret

00B6'             de_sel:
00B6' 3E 00       ld      a,0          :deselect simulator
00B8' 32 1777*    ld      (selflg),a

00BB' C9          ret

;*****
;*
;* Subroutine :      Status
;* Function :      Send buffer status to host
;*               computer
;* Inputs :      Bufcnt = # bytes in buffer
;* Outputs :      none
;*
;*****

00BC'             status:
00BC' FB          ei              :enable burst interrupts while sending status

00BD' 2A 0004*    ld      hl,(bufcnt)  :get buffer byte count

00C0' 7D          ld      a,l          :& send it
00C1' CD 0169'    call  send
00C4' 7C          ld      a,h
00C5' CD 0169'    call  send

00C8' C9          ret

;*****
;*
;* Subroutine :      Databytes
;* Function :      process databytes
;* Inputs :      b = byte just received
;*               dataexp = 0 --> data command
;*               dataexp = 5 --> station number
;*               :      dataexp = 1..4 --> databyte
;* Outputs :      dataexp updated
;*               databuffer updated
;*
;*****

00C9'             databytes:

; process data command
00C9' 3A 1778*    ld      a,(dataexp)  : what to do?
00CC' FE 00       cp      0            : process datacommand (ldh)?
00CE' C2 00D9'    jp      nz, l1

```

MACRO-80 3.44 09-Dec-81

PAGE 1-5

```

00D1' 3E 05          1d      a,5          ; yes --> 5 more databytes to be expected
00D3' 32 1778*       1d      (dataexp),a
00D6' C3 0127'       jp      da_ret
00D9'                11:
00D9' 3A 1777*       1d      a,(selflg)    ; simulator selected ?
00DC' 87             or      a
00DD' C2 00EA'       jp      nz,18        ; ---> databytes should be processed
00E0' 3A 1778*       1d      a,(dataexp)    ; else, databytes should only be counted
00E3' 3D            dec      a
00E4' 32 1778*       1d      (dataexp),a
00E7' C3 0127'       jp      da_ret

00EA'                18:      ; process station number
00EA' FE 05          cp      5            ; station number received?
00EC' C2 00FB'       jp      nz,12
00EF' 78            1d      a,b          ; get station number for following databytes
00F0' 32 1779*       1d      (station), a
00F3' 3E 04          1d      a,4
00F5' 32 1778*       1d      (dataexp),a    ; 4 databytes to be expected
00F8' C3 0127'       jp      da_ret
00FB'                12:

; write data byte
00FB' 21 0000*       1d      hl,datwr     ; find appropriate write pointer
00FE' 3A 1779*       1d      a,(station)
0101' 16 00          1d      d,0
0103' 5F            1d      e,a
0104' 19            add     hl,de
0105' E5            push    hl           ; save pointer to write index
0106' 4E            1d      c,(hl)       ; reg. c contains write index

0107' 07            rlca                ; multiply stationnumber with 16
0108' 07            rlca
0109' 07            rlca
010A' 07            rlca
010B' 5F            1d      e,a          ; move to bc
010C' 16 00          1d      d,0
010E' 21 0000*       1d      hl,datbuf##
0111' 19            add     hl,de        ; find stationoffset in datbuf
0112' 59            1d      e,c          ; add write index
0113' 19            add     hl,de

0114' 70            1d      (hl),b       ; write received data byte

; update write index
0115' E1            pop     hl           ; restore pointer to write index
0116' 0C            inc     c
0117' 79            1d      a,c          ; end of buffer reached?
0118' FE 10          cp      16
011A' C2 011F'       jp      nz, 13
011D' 0E 00          1d      c,0         ; wrap around
011F' 71            13: 1d      (hl), c

0120' 3A 1778*       1d      a, (dataexp)
0123' 3D            dec      a
0124' 32 1778*       1d      (dataexp), a

```


MACRO-B0 3.44 09-Dec-81 PAGE 1-6

```

0127' C9          da_ret: ret

;*****
;*
;*      Subroutine :   Buf_read
;*      Function :    Read character from input
;*      buffer
;*      Inputs :      RDPNT = source pointer
;*                   BUFCNT = byte count
;*      Outputs :     A = character from buffer
;*
;*****

0128'          buf_read:
0128' E5          push    hl
0129' 2A 0000'     ld      hl,(rdpnt)    ;get pointer
012C' 7E          ld      a,(hl)        ;get character
012D' F5          push    af            ;& save it

012E' 23          inc     hl            ;update pointer
                                cmphl   buffer+bsize ;past end of buffer ?
012F' 7C          +      ld      a,h      ;compare HL with <val>
0130' FE 17'      +      cp      high (buffer+bsize)
0132' 20 03      +      jr      nz,...0003
0134' 7D          +      ld      a,l
0135' FE 76'      +      cp      low (buffer+bsize)
0137'          +      ..0003:
0137' 20 03      jr      nz,rd_ok        ;no, continue
0139' 21 0006'   ld      hl,buffer      ;wrap around
013C'          rd_ok:
013C' 22 0000'   ld      (rdpnt),hl     ;save pointer

013F' F3          di                    ;no interrupts here
0140' 2A 0004'   ld      hl,(bufcnt)    ;decrement byte count
0143' 2B          dec     hl
0144' 22 0004'   ld      (bufcnt),hl
0147' FB          ei

                                cmphl   bstart    ;lower threshold reached ?
0148' 7C          +      ld      a,h      ;compare HL with <val>
0149' FE 13      +      cp      high (bstart)
014B' 20 03      +      jr      nz,...0004
014D' 7D          +      ld      a,l
014E' FE 88      +      cp      low (bstart)
0150'          +      ..0004:
0150' 20 04      jr      nz,not_en

0152' 3E 27      ld      a,00100111b   ;enable host transmitter ; DTR high
0154' D3 91      out     (uartc),a
0156'          not_en:
0156' F1          pop     af            ;restore character & return
0157' E1          pop     hl
0158' C9          ret

```

MACRO-80 3.44 09-Dec-81 PAGE 1-7

```

*****
;*
;* Subroutine :   RX_stat
;* Function :   get input buffer status
;* Inputs :     BUFCNT = byte count
;* Outputs :    zero flag : NZ if character
;*              available in buffer
;*
*****

```

```

0159' rx_stat::
0159' E5      push    hl
015A' 2A 0004' ld      hl,(bufcnt) ;get byte count
015D' 7C      ld      a,h
015E' B5      or      l           ;set zero flag
015F' E1      pop     hl
0160' C9      ret

```

```

*****
;*
;* Subroutine :   Get_byt
;* Function :     get character from buffer
;* Inputs :       BUFCNT = byte count
;*               data in buffer
;* Outputs :      A = character
;*
*****

```

```

0161' get_byt::
0161' CD 0159' call    rx_stat      ;get buffer status
0164' 28 FB      jr      z,get_byt ;wait until character available
0166' C3 0128' jp      buf_read  ;get character & return

```

```

*****
;*
;* subroutine :   Send
;* function :     send character to host computer
;* inputs :      a = character
;* outputs :     none
;*
*****

```

```

0169' send::
0169' F5      push    af

016A' 3A 1777' ld      a,(selflg) ; sending is only allowed if simulator
016D' 87      or      a           ; is selected
016E' C2 0173' jp      nz,sendlp ;
0171' F1      pop     af
0172' C9      ret

```

```

0173' sendlp:
0173' DB 91      in      a,(uartc) ;uart ready ?
0175' E6 01      and     1

```

```

MACRO-80 3.44    09-Dec-81    PAGE    1-8

0177'  28 FA                jr      z,sendlp

0179'  F1                   pop     af
017A'  03 90                out     (uartd),a      ;send character

017C'  C9                   ret

;***** data area *****

017D'                dseg

0000*                rdpnt: ds      2            ;buffer read pointer
0002*                wrpnt: ds      2            ;buffer write pointer
0004*                bufcnt: ds      2            ;buffer byte count
0006*                buffer: ds      bsize        ;data buffer
1776*                scode: ds      1            ;simulator select code
1777*                selflg: ds      1            ;simulator select flag

1778*                dataexp:ds      1            ;how many databytes must we expect
1779*                station:ds      1            ;for which station are we receiving databytes?

177A'  00                db      0
                                end

```


MACRO-80 3.44 09-Dec-81 PAGE 5

Macros:

CMPHL COMP16

Symbols:

007D' ..0000	008A' ..0001	00A0' ..0002
0137' ..0003	0150' ..0004	1770 BSIZ
1388 BSTART	157C BSTOP	0004* BUFENT
0006* BUFFER	0128* BUF_READ	00C9* DATABYTES
001A DATACMD	1778* DATAEXP	010F* DATBUF
0025* DATRD	00FC* DATWR	0127* DA_RET
0086* DE_SEL	0161I' GET_BYT	0000I' INITRS
0009* L1	00FB' L2	011F' L3
002C' L4	00EA' L8	0156' NOT_EN
0000* RDPNT	013C' RD_OK	0090* RX_CONT
0047I' RX_INT	0159I' RX_STAT	1776* SCODE
001E SEL1	001F SEL2	00A9* SELECT
1777* SELFLG	003E* SEL_OK	0169I' SEND
0173* SENDLP	003C* SET_2	0070* START
001C STATCMD	1779* STATION	00BC* STATUS
0019 STRTCMD	00A0 SWPORT	0091 UARTC
0090 UARTD	0002* WRPNT	00A5* WR_OK

No Fatal error(s)

□

MACRO-80 3.44 09-Dec-81 PAGE 1

```

; /*****
;*
;*      SIMBRS.MAC      Burst Interrupt processing routines
;*
;*
;*      Version          : 0
;*      Last modification : 15/02/1993
;*      Author           : A.M. Noorbergen
;*
;*      Modified version of SIMBRS.MAC written by R. Kellenbach
;*      (C) Delft University of Technology, 1987, 1993
;*
;*      17/02/1993 : Eurofix mode can be set for each station separately
;*
; \*****/

```

```

0000      timer0 equ 00h      ;8253 timers
0001      timer1 equ 01h
0002      timer2 equ 02h
0003      timmod equ 03h      ;8253 mode register

0004      gnddel equ 04h      ;gndwave delay port
0008      skydel equ 08h      ;skywave delay port

000C      delport equ 0ch      ;delay setting port (fine delay steps)

0010      gndecd equ 10h      ;gndwave ECD port
0014      skyecd equ 14h      ;skywave ECD-port

0018      dacdata equ 18h      ;LOGDAC data port
001C      daccont equ 1ch      ;LOGDAC control port

0080      intcl1 equ 80h      ;interrupt clear flip-flop's
0081      intcl2 equ 81h

0082      trig1 equ 82h      ;scope trigger ports
0083      trig2 equ 83h
0084      trig3 equ 84h
0085      trig4 equ 85h

0016      SYN equ 16h      ;ASCII SYN character

      .z80

0000*      cseg

      cmph1 macro val      ;16 bit compare macro
      local cmprdy
      ld a,h      ;compare HL with <val>
      cp high (val)
      jr nz,cmprdy

      ld a,l
      cp low (val)

      cmprdy:

```


MACRO-80 3.44 09-Dec-81 PAGE 1-1

endm

```

comp16 macro                ; 16 bit compare macro
    ld    a,h                ; compare HL and DE
    cp    d                    ; zero flag set when equal
    jr    nz,co_rdy
    ld    a,1
    cp    e
co_rdy:
endm

```

```

*****
;*                               *
;* Subroutine :   Initbrs         *
;* Function  :   Initialize variables and *
;*                               *
;*                               *
;* hardware for burst interrupts *
;*                               *
;* Inputs   :   none              *
;*                               *
;* Outputs  :   misc. variables initialized *
;*                               *
;*                               *
*****

```

```

0000*      initbrs::
0000*      3E 00          ld a,0                ;disable interrupt flip-flop's
0002*      D3 80          out (intc11),a
0004*      3E 34          ld a,00110100b        ;init timer 0
0006*      D3 03          out (timmod),a        ;mode 2 (rate generator)
0008*      3E 32          ld a,50                ;generate 100 kHz pulse
000A*      D3 00          out (timer0),a
000C*      3E 00          ld a,0
000E*      D3 00          out (timer0),a
0010*      3E 74          ld a,01110100b        ;init timer 1
0012*      D3 03          out (timmod),a        ;mode 2 (rate generator)
0014*      3E BA          ld a,10111010b        ;init timer 2
0016*      D3 03          out (timmod),a        ;mode 5 (hardware triggered strobe)
0018*      3E 00          ld a,0                ;select prom area for ECD
001A*      D3 10          out (gndec),a
001C*      32 0015*      ld (g_ecd),a
001F*      D3 14          out (skyecd),a
0021*      32 0016*      ld (s_ecd),a
0024*      3E 00          ld a,0                ;init delay PAL's
0026*      D3 04          out (gnddel),a
0028*      D3 08          out (skydel),a
002A*      3E 0F          ld a,00001111b        ;init LOGDAC control port, disable DAC's
002C*      D3 1C          out (daccont),a

```

MACRO-80 3.44 09-Dec-81

PAGE 1-2

```

002E* 3E 00          ld a,0          ;enable trig3 pulses at every interrupt
0030* 03 84          out (trig3),a

0032* 3E 00          ld a,0          ;index = 0
0034* 32 0000*       ld (index),a

0037* 3E 07          ld a,7          ;brsct = 7
0039* 32 0002*       ld (brsct),a

003C* 3E 00          ld a,0          ;set sign bit
003E* 32 0017*       ld (sign),a

0041* 3E 03          ld a,3          ;3 stations
0043* 32 0001*       ld (nstat),a

0046* 21 0000        ld hl,0          ;remainder = 0
0049* 22 000D*       ld (remain),hl

004C* 21 00DD*       ld hl,phcinit    ;init phase codes
004F* 11 0004*       ld de,phctab
0052* 01 0007        ld bc,7
0055* ED B0          ldir

0057* 3A 0004*       ld a,(phctab)    ;phase code = 1st master code
005A* 32 0003*       ld (phcode),a

005D* 21 00E4*       ld hl,dtainit    ;init dta's
0060* 11 0019*       ld de,dtatab
0063* 01 0015        ld bc,3*7
0066* ED B0          ldir

0068* 21 00F9*       ld hl,skyinit    ;init skywave delay's
006B* 11 0035*       ld de,skytb
006E* 01 000E        ld bc,2*7
0071* ED B0          ldir

0073* 21 0107*       ld hl,g_levinit  ;init gndwave levels
0076* 11 0051*       ld de,gnd_lev
0079* 01 0007        ld bc,7
007C* ED B0          ldir

007E* 21 010E*       ld hl,s_levinit  ;init skywave levels
0081* 11 0058*       ld de,sky_lev
0084* 01 0007        ld bc,7
0087* ED B0          ldir

0089* 21 0115*       ld hl,ecdinit    ;init gndwave ECD's
008C* 11 005F*       ld de,g_ecdtb
008F* 01 0007        ld bc,7
0092* ED B0          ldir

0094* 21 0115*       ld hl,ecdinit    ;init skywave ECD's
0097* 11 0066*       ld de,s_ecdtb
009A* 01 0007        ld bc,7
009D* ED B0          ldir

```


MACRO-80 3.44 09-Dec-81

PAGE 1-3

```

009F' 21 011C'      ld hl,mov_init      ;clear move flags
00A2' 11 006D"      ld de,mov_flg
00A5' 01 0007      ld bc,7
00A8' ED B0        ldir

00AA' 3E FF        ld a,255              ;no noise
00AC' 32 0018"      ld (noise),a

00AF' 21 0123'      ld hl, patt_init      ;init Eurofix patterns
00B2' 11 007B"      ld de, patt_0
00B5' 01 0002      ld bc, 2
00B8' ED B0        ldir

00BA' 21 0125'      ld hl, shift_init      ;init pos/neg Eurofix shifts
00BD' 11 007D"      ld de, shift_0
00C0' 01 0002      ld bc, 2
00C3' ED B0        ldir

00C5' 3A 0127'      ld a,(adv_init)        ;init Eurofix burst advance time
00C8' 32 007F"      ld (adv_tim),a

00CB' 3A 0128'      ld a,(del_init)        ;init Eurofix burst delay time
00CE' 32 0080"      ld (del_tim),a

00D1' 21 0129'      ld hl,mode_init      ;init operation mode: no eurofix
00D4' 11 0081"      ld de,mod_tab
00D7' 01 0007      ld bc,7
00DA' ED B0        ldir

00DC' C9           ret

;phcinit:db 0FFh,0FFh,0FFh,0FFh,0FFh,0FFh,0FFh ;phase codes for testing
00DD' CA F9 F9 F9   phcinit:db 0CAh,0F9h,0F9h,0F9h,0F9h,0F9h,0F9h ;initial phase codes
00E1' F9 F9 F9

00E4' A0 86 01      dtainit:db 160,134,1 ;2.5 ms
00E7' C0 D4 01      db 192,212,1 ;3.0 ms
00EA' E0 22 02      db 224,34,2 ;3.5 ms
                    rept 4
                    db 0,0,0
                    endm

00ED' 00 00 00      + db 0,0,0
00F0' 00 00 00      + db 0,0,0
00F3' 00 00 00      + db 0,0,0
00F6' 00 00 00      + db 0,0,0

00F9' 0320 0384      skyinit:dword 800,900,1000,0,0,0,0 ;20, 22.5, 25 us
00FD' 03E8 0000
0101' 0000 0000
0105' 0000

0107' 00 10 35 00    g_levinit:db 0,16,53,0,0,0,0 ;0, -6, -20 dB
010B' 00 00 00
010E' FF FF FF FF    s_levinit:db 255,255,255,255,255,255,255 ;no skywaves
0112' FF FF FF

```

MACRO-80 3.44 09-Dec-81 PAGE 1-4

```

0115' 0A 0A 0A 0A      ecdinit:db  10,10,10,10,10,10      ;initial ECD's : 0 us
0119' 0A 0A 0A

011C' 00 00 00 00      mov_init:db  0,0,0,0,0,0,0
0120' 00 00 00

0123' 3F                patt_init:db  00111111b           ;pattern for 0 databit
0124' 3F                db 00111111b           ;pattern for 1 databit

0125' 2A                shft_init:db  00101010b           ;pos/neg shifts for 0
0126' 15                db 00010101b           ;pos/neg shifts for 1

0127' 28                adv_init: db  40                ;40 units of 25 ns
0128' 28                del_init: db  40                ;40 units of 25 ns

0129' 00 00 00 00      mode_init:db  0,0,0,0,0,0,0       ;Initial mode : no Eurofix
012D' 00 00 00

;*****
;*
;*      Subroutine :      Start
;*      Function :      start burst generating process
;*      Inputs :      none
;*      Outputs :      none
;*
;*****

0130'      start::

0130' 3A 0001"          ld  a,(nstat)      ; fool "prep_next" to let it prepare
0133' 3D                dec  a              ; the system for the master. Therefore
0134' 4F                ld  c,a            ; "prep_next" has to think that the current
0135' 06 00             ld  b,0            ; station is (nstat-1)
0137' 3E FF            ld  a,0ffh
0139' 32 011B"          ld  (dptrd),a
013C' 3E 01            ld  a,1
013E' 32 009D"          ld  (data_cnt),a
0141' CD 03BC'          call prep_next

;the first burst of the master is going to be lost,
;so everything has to be set ready for the second burst of
;the master. This means that the variables "patt_tab", "shft_tab"
;and "shft_cnt" have to be set accordingly

0144' 3A 0088"          ld  a,(patt_tab)  ;correct patt_tab for lost first burst
0147' 07                rlc a
0148' 32 0088"          ld  (patt_tab),a
014B' 3A 008F"          ld  a,(shft_tab)  ;correct shft_tab for lost first burst
014E' 07                rlc a
014F' 32 008F"          ld  (shft_tab),a

0152' 01 0000          ld  bc,0
0155' 3E 00            ld  a,0

```


MACRO-80 3.44 09-Dec-81 PAGE 1-5

```

0157* 32 009E*      ld      (nxt_index),a
015A* 3E FF         ld      a,-1
015C* 32 0002*      ld      (brsct),a

015F* 21 0000      ld      hl,0          ; initialize remainder, skywave delay
0162* 22 000D*      ld      (remain),hl   ; timer and delay lines
0165* 22 0011*      ld      (temprem),hl
0168* CD 031C*      call    set_del

016B* 3A 0003*      ld      a,(phcode)
016E* 07            rlca
016F* 32 0003*      ld      (phcode),a

0172* 21 0064      ld      hl,100
0175* 7D            ld      a,l
0176* D3 01        out     (timer1),a
0178* 7C            ld      a,h
0179* D3 01        out     (timer1),a

017B* CD 01CB*      call    set_timer

017E* CD 04A9*      call    set_lev

0181* CD 04F2*      call    carriers_on

0184* 3E 06        ld      a,6
0186* 32 0002*      ld      (brsct),a
0189* 3E 00        ld      a,0
018B* 32 0000*      ld      (index),a

018E* 3E 00        ld      a,0          ;clear interrupt flip-flop 1
0190* D3 80        out     (intc11),a
0192* 3E 01        ld      a,1          ;enable interrupts
0194* D3 80        out     (intc11),a

0196* C9            ret

```

```

*****
;*                                     *
;*      Subroutine :   Brsint          *
;*      Function   :   Burst timer interrupt processing *
;*      Inputs    :   misc. variables *
;*      Outputs   :   misc. variables *
;*                                     *
*****

```

```

0197*      brs_int::

0197* 3E 38        ld      a,56          ; 280 microsec. to wait after
0199* 32 009F*      ld      (timeleft),a ; start of burst

019C* 3E 00        ld      a,0          ;clear interrupt flip-flops
019E* D3 80        out     (intc11),a

```



```

MACRO:80 3.44 09-Dec-81 PAGE 1-6

01A0' 3E 01          ld    a,1
01A2' 03 80          out   (intc11),a

01A4' FB            ei             ;enable RS-232

01A5' 3E 01          ld    a,1          ;disable trigger 1 & 2 pulses
01A7' 03 82          out   (trig1),a
01A9' 03 83          out   (trig2),a

01AB' 3A 0000"       ld    a,(index)    ;bc = station index
01AE' 4F             ld    c,a
01AF' 06 00          ld    b,0

01B1' 21 0002"       ld    hl,brscnt    ;decrement burst counter
01B4' 35             dec    (hl)

01B5' GC 03BC"       call  z,prep_next   ;prepare for next station

01B8' FA 0458"       jp    m,next_stat   ;switch to next station

01BB' 2A 0011"       ld    hl,(temprem)  ;save temprem for "set_del"
01BE' 22 0013"       ld    (temprem2),hl

01C1' CD 01CB"       call  set_timer     ;calculate new quotient, tempquo,
                                         ;remainder and temprem for burst after the
                                         ;next one

01C4' CD 031C"       call  set_del       ;set delay lines

01C7' CD 04F2"       call  carriers_on   ;switch Loran-C carriers on

01CA' C9             ret

;*****
;*
;* Subroutine : Set_timer
;* Function : prepare timer for next 1 ms +/- interval with
;*           with Eurofix modulation
;* Inputs : bc = index of current station
;* Outputs : misc. variables
;*
;*****

01CB' set_timer:
01CB' C5             push  bc             ;determine whether timer has to be set
01CC' 3A 0002"       ld    a,(brscnt)    ;to inter-burst interval (+/- 1 ms) or
01CF' B7             or     a             ;to inter-station interval (dta)
01D0' CA 01D9"       jp    z,next_dta     ;next DTA has to be programmed into 8253
01D3' FA 0223"       jp    m,inter_stat   ;new 1 ms value has to be programmed
01D6' C3 0232"       jp    119

01D9' next_dta:
                                         ; during the burst before the last, (quotient) might have been
                                         ; set at a value, different from 100. This has to be taken into

```

MACRO-80 3.44 09-Dec-81

PAGE 1-7

; account here.

```

0109' F3          di          ; the subroutine "set_timer" will take
010A' 3A 009F'    ld          a,(timeleft) ; about 321 states before reaching 119
010D' 06 0C      sub          12          ; so "timeleft" can be decreased 12 units
010F' 32 009F'    ld          (timeleft),a ; of 25 states
01E2' FB          ei

01E3' 2A 000B'    ld          hl,(quotient)
01E6' 11 0064     ld          de,100
01E9' 37          scf
01EA' 3F          ccf
01EB' ED 52      sbc          hl,de
01ED' E5          push        hl          ;this result will be used later on

01EE' 21 0019'    ld          hl,dtatab   ; get quotient and remainder for next DTA
01F1' 09          add         hl,bc
01F2' 09          add         hl,bc
01F3' 09          add         hl,bc
01F4' 09          add         hl,bc
01F5' 5E          ld          e,(hl)
01F6' 23          inc         hl
01F7' 56          ld          d,(hl)      ; DE now contains quotient
01F8' 23          inc         hl
01F9' 4E          ld          c,(hl)
01FA' 23          inc         hl
01FB' 46          ld          b,(hl)      ; BC now contains remainder

01FC' E1          pop         hl          ; adjust quotient
01FD' 19          add         hl,de
01FE' EB          ex          de,hl      ; DE now contains corrected quotient

01FF' 2A 000D'    ld          hl,(remain)
0202' 09          add         hl,bc
cmphl 400
0203' 7C          +          ld          a,h          ;compare HL with <val>
0204' FE 01      +          cp          high (400)
0206' 20 03      +          jr          nz,...0000
0208' 7D          +          ld          a,l
0209' FE 90      +          cp          low (400)
020B'             +          ..0000:
020B' 38 05      jr          c,rm_ok2
020D' 01 FE70     ld          bc,-400
0210' 09          add         hl,bc
0211' 13          inc         de
0212' 22 000D'    rm_ok2: ld          (remain),hl
0215' ED 53 000B' ld          (quotient),de
0219' 22 0011'    ld          (temprem),hl
021C' ED 53 000F' ld          (tempquo),de
0220' C3 0232'    jp          119

0223'             inter_stat: ;set timer to 1 ms
0223' 2A 000D'    ld          hl,(remain)
0226' 22 0011'    ld          (temprem),hl
0229' 21 0064     ld          hl,100
022C' 22 000B'    ld          (quotient),hl

```


MACRO-80 3.44 09-Dec-81 PAGE 1-8

```

022F' 22 000F'      ld      (tempquo),hl

0232'               119:      ; at this point (quotient) + (remain) contain the timer and
                        ; delay line settings for "normal" operation. For Eurofix these
                        ; values may have to be corrected with approx. 1 us plus or
                        ; minus. After that, the 8253 will be set right away, the delay
                        ; lines will be set in the next interruptcycle

0232' 21 0081'      ld      hl,mod_tab      ;get operation mode for station and
0235' 3A 009E'      ld      a,(nxt_index)    ;determine whether to "eurofix" or not
0238' 4F             ld      c,a
0239' 06 00         ld      b,0
023B' 09            add     hl,bc
023C' 7E            ld      a,(hl)           ;mode 0 means no eurofix:
023D' B7            or      a               ;no burst shifts necessary, so skip
023E' CA 0311'      jp      z,1110          ;the following code

0241' 21 0088'      ld      hl,patt_tab     ;find out (via pattern) whether burst
0244' 3A 009E'      ld      a,(nxt_index)    ;has to be shifted or not
0247' 4F             ld      c,a
0248' 06 00         ld      b,0
024A' 09            add     hl,bc           ;BC still contains (nxt_index)
024B' 7E            ld      a,(hl)
024C' 07            rlca                    ;shift modulation pattern to left
024D' 77            ld      (hl),a
024E' D2 02F2'      jp      nc,noshift      ;OK, no burst shifting

0251' F3            di                     ; the subroutine "set_timer" will take
0252' 3A 009F'      ld      a,(timeleft)     ; about 586 states following this path
0255' D6 14         sub     20              ; so "timeleft" can be decreased 20 units
0257' 32 009F'      ld      (timeleft),a    ; of 25 states
025A' FB            ei

025B' 21 008F'      ld      hl,shft_tab     ;burst shifting -> what way?
025E' 09            add     hl,bc
025F' 7E            ld      a,(hl)
0260' 07            rlca
0261' 77            ld      (hl),a
0262' DA 02AB'      jp      c,negshft       ;negative?

0265' 3A 0080'      ld      a,(del_tim)     ;positive shift: add extra delay
0268' ED 5B 000B'   ld      de,(quotient)
026C' 4F             ld      c,a            ;BC contains burst delay in 25 ns units
026D' 06 00         ld      b,0

026F' 2A 000D'      ld      hl,(remain)     ;add extra delay to (tempquo) and (temprem)
0272' 09            add     hl,bc
                        cmphl 400
0273' 7C            +      ld      a,h       ;compare HL with <val>
0274' FE 01         +      cp      high(400)
0276' 20 03         +      jr      nz,...0001
0278' 7D            +      ld      a,l
0279' FE 90         +      cp      low(400)
027B'              +      ..0001:
027B' 38 05         jr      c,rm_ok3

```

MACRO-80 3.44 09-Dec-81 PAGE 1-9

```

027D' 01 FE70      1d      bc,-400
0280' 09           add     hl,bc
0281' 13           inc     de
0282' 22 0011"     rm_ok3: 1d      (temprem),hl
0285' ED 53 000F"  1d      (tempquo),de

0289' 3A 0080"     1d      a,(del_tim)
028C' 4F           1d      c,a
028D' 06 00       1d      b,0
028F' 11 0064     1d      de,100
0292' 2A 0011"     1d      hl,(temprem)
0295' 37           scf
0296' 3F           ccf
0297' ED 42       sbc      hl,bc
0299' D2 02A1'    jp       nc,rm_ok9
029C' 01 0190     1d      bc,400
029F' 09           add     hl,bc
02A0' 1B           dec     de
02A1' 22 000D"     rm_ok9: 1d      (remain),hl
02A4' ED 53 000B"  1d      (quotient),de
02A8' C3 0311'    jp       1110

02AB' 3A 007F"     negshft:1d      a,(adv_tim)      ;burst advance time for neg. shift
02AE' 4F           1d      c,a
02AF' 06 00       1d      b,0
02B1' ED 5B 000B"  1d      de,(quotient)
02B5' 37           scf      ; reset carry flag
02B6' 3F           ccf
02B7' 2A 000D"     1d      hl,(remain)
02BA' ED 42       sbc      hl,bc
02BC' D2 02C4'    jp       nc,rm_ok4
02BF' 01 0190     1d      bc,400
02C2' 09           add     hl,bc
02C3' 1B           dec     de
02C4' 22 0011"     rm_ok4: 1d      (temprem),hl
02C7' ED 53 000F"  1d      (tempquo),de

02CB' 3A 007F"     1d      a,(adv_tim)
02CF' 4F           1d      c,a
02CF' 06 00       1d      b,0
02D1' 11 0064     1d      de,100
02D4' 2A 0011"     1d      hl,(temprem)
02D7' 09           add     hl,bc
                        cmphl 400
02D8' 7C           +      1d      a,h      ;compare HL with <val>
02D9' FE 01       +      cp      high (400)
02DB' 20 03       +      jr      nz,...0002
02DD' 7D           +      1d      a,l
02DE' FE 90       +      cp      low (400)
02E0'             +      ...0002:
02E0' DA 02E8'    jp       c,rm_ok8
02E3' 01 FE70     1d      bc,-400
02E6' 09           add     hl,bc
02E7' 13           inc     de
02E8' 22 000D"     rm_ok8: 1d      (remain),hl
02EB' ED 53 000B"  1d      (quotient),de

```


MACRO-B0 3.44 09-Dec-81 PAGE 1-10

```

02EF' C3 0311'      jp      1110

02F2'              noshift:          ;This burst stays, but do shift (shift_tab)
02F2' 21 008F'      ld      hl,shift_tab ;find out (via pattern) whether burst
02F5' 3A 009E'      ld      a,(nxt_index) ;has to be shifted or not
02F8' 4F            ld      c,a
02F9' 06 00        ld      b,0
02FB' 09            add     hl,bc
02FC' 7E            ld      a,(hl)
02FD' 07            rlc     a          ;shift shift pattern to left
02FE' 77            ld      (hl),a
02FF' 2A 000D'     ld      hl,(remain)
0302' 22 0011'     ld      (temprem),hl
0305' 2A 000B'     ld      hl,(quotient)
0308' 22 000F'     ld      (tempquo),hl
030B' 21 0064      ld      hl,100
030E' 22 000B'     ld      (quotient),hl

;set timer according to values in (tempquo)
;the delay lines will be set in the next interrupt cycle

0311' 2A 000F'     1110: ld      hl,(tempquo)
0314' 7D            ld      a,l
0315' 03 01        out     (timer1),a
0317' 7C            ld      a,h
0318' D3 01        out     (timer1),a

031A' C1            pop     bc

031B' C9            ret

```

```

;*****
;*
;* Subroutine : Set_del
;* Function : set gndwave & skywave delay ports,
;*           rotate phase code for next burst
;* Inputs : remain = gndwave DTA fraction
;*          data in skytab
;*          BC = index for next station
;* Outputs : Carriers are switched off!!
;*
;*****

```

```

031C' C5            set_del:
031C' C5            push     bc

031D' 3A 0002'     ld      a,(brscent)
0320' B7            or      a
0321' FA 032D'     jp      m,1111

```

MACRO-80 3.44 09-Dec-81

PAGE 1-11

```

0324' 3A 0000"      ld      a,(index)
0327' 06 00         ld      b,0
0329' 4F            ld      c,a
032A' C3 0333'      jp      1112
032D' 3A 009E"      1111: ld      a,(nxt_index)
0330' 06 00         ld      b,0
0332' 4F            ld      c,a

0333'              1112:

0333' 2A 0013"      ld      hl,(temprem2) ;get remainder of DTA
                        rept 3 ;convert to 200 ns units
                        srl     h
                        rr      l ;divide HL by 8
                        endm
0336' CB 3C          +      srl     h
0338' CB 1D          +      rr      l ;divide HL by 8
033A' CB 3C          +      srl     h
033C' CB 1D          +      rr      l ;divide HL by 8
033E' CB 3C          +      srl     h
0340' CB 1D          +      rr      l ;divide HL by 8

0342' 3E 31         ld      a,49
0344' 95            sub      1
0345' 32 00A0"      ld      (store1),a ;save result

0348' 21 0035"      ld      hl,skytob
034B' 09            add      hl,bc
034C' 09            add      hl,bc
034D' 09            add      hl,bc
034E' 09            add      hl,bc

034F' 5E            ld      e,(hl) ;DE = quotient part (= 10us units)
0350' 23            inc     hl
0351' 56            ld      d,(hl)
0352' 23            inc     hl
0353' C5            push    bc
0354' 4E            ld      c,(hl)
0355' 23            inc     hl
0356' 46            ld      b,(hl) ;BC = remainder part (= 25ns units)

0357' 2A 0013"      ld      hl,(temprem2) ;add remainder
035A' 09            add      hl,bc
                        cmphl 400
035B' 7C          +      ld      a,h ;compare HL with <val>
035C' FE 01          +      cp      high (400)
035E' 20 03          +      jr      nz,...0003
0360' 7D          +      ld      a,l
0361' FE 90          +      cp      low (400)
0363'              +      ..0003:
0363' DA 036B'      jp      c,rm_ok5
0366' 01 FE70       ld      bc,-400
0369' 09            add      hl,bc
036A' 13            inc     de

```


MACRO-80 3.44 09-Dec-81 PAGE 1-12

```

036B'  C1          rm_ok5: pop    bc
036C'  1B          dec    de          ;decrement timer count
                                         ;(timer needs 1 clock pulse to trigger)

036D'  7B          ld      a,e          ;set skywave delay timer
036E'  03 02      out     (timer2),a
0370'  7A          ld      a,d
0371'  03 02      out     (timer2),a

0373'  E5          push   hl          ;save skywave delay fraction
                        rept    3
                        srl    h          ;convert to 200 ns units
                        rr     l
                        endm

0374'  CB 3C      +      srl    h          ;convert to 200 ns units
0375'  CB 10      +      rr     l
0378'  CB 3C      +      srl    h          ;convert to 200 ns units
037A'  CB 10      +      rr     l
037C'  CB 3C      +      srl    h          ;convert to 200 ns units
037E'  CB 10      +      rr     l

0380'  3E 31      ld      a,49          ;set skywave delay PAL
0382'  95          sub     1
0383'  32 00A2*   ld      (store2),a

0386'  E1          pop     hl
0387'  7D          ld      a,l          ;get 25 ns units of skywave delay
0388'  E6 07      and     7
038A'  CB 27      sla     a          ;shift into position
038C'  CB 27      sla     a
038E'  CB 27      sla     a
0390'  5F          ld      e,a

0391'  2A 0013*   ld      hl,(temprem2) ;get 25 units of gndwave delay
0394'  7D          ld      a,l
0395'  E6 07      and     7
0397'  83          or      e          ;add skywave bits
0398'  03 0C      out     (delport),a ;set delay lines

039A'  3A 009F*   ld      a,(timeleft)
039D'  87          or      a
039E'  FA 03AA*   jp      m,1130      ;more than 300 us have passed since
                                         ;begin of burst, so we can switch off
                                         ;carriers safely

03A1'  26 00      ld      h,0          ;wait until 300 us have passed since
03A3'  7D          ld      a,l          ;begin of burst
03A4'  2B          delay: dec    hl          ;gndwave+skywave have faded out by then
03A5'  7C          ld      a,h
03A6'  85          or      l
03A7'  C2 03A4*   jp      nz,delay

03AA'  3E 40      1130: ld      a,01000000b ;switch carriers off
03AC'  03 10      out     (gndec),a
03AE'  03 14      out     (skyecd),a

```

MACRO-80 3.44 09-Dec-81 PAGE 1-13

```

03B0' 3A 00A0'      1d    a,(store1)      ;set delay lines
03B3' 03 04          out    (gnddel),a

03B5' 3A 00A2'      1d    a,(store2)
03B8' 03 08          out    (skydel),a

03BA' C1            pop    bc

03BB' C9            ret                      ;carriers are switched off this very moment

```

```

;*****
;*
;*      Subroutine :   Prep_next
;*      Function  :   Prepare for next station
;*      Inputs   :   bc = index of current station
;*      Outputs  :   (nxt_index) = index of following stat. in chain
;*                  new modulation pattern will be set ready
;*
;*
;*****

```

```

03BC'      prep_next:
03BC' C5          push    bc                ; bc will be used as next index further on

03BD' 3A 0001'      1d    a,(nstat)
03C0' 0C          inc    c
03C1' B9          cp     c                ; is current station last station in chain?
03C2' CA 03C9'      jp     z, first        ; yes --> next station will have index 0
03C5' 79          1d    a,c
03C6' C3 03CD'      jp     115
03C9' 3E 00          first: 1d    a,0
03CB' 0E 00          1d    c,0
03CD' 32 009E'      115: 1d    (nxt_index),a

03D0' 21 00B1'      1d    hl,mod_tab      ;get operation mode for station and
03D3' 09          add    hl,bc            ;determine whether to "eurofix" or not
03D4' 7E          1d    a,(hl)           ;mode 0 means no eurofix:
03D5' B7          or     a                ;no burst shifts necessary, so skip
03D6' CA 044C'      jp     z,nxt_ret      ;the following code

03D9' 3A 009E'      1d    a,(nxt_index)   ; if current station is last in chain
03DC' B7          or     a                ; (so next station is master again), then
03DD' C2 0407'      jp     nz,new_byte    ; a new byte might be necessary

03E0' 3A 009D'      1d    a,(data_cnt)    ;
03E3' B7          or     a
03E4' C2 03EF'      jp     nz,1120
03E7' 3E 07          1d    a,7
03E9' 32 009D'      1d    (data_cnt),a
03EC' C3 0407'      jp     new_byte

03EF' 3D          1120: dec    a
03F0' 32 009D'      1d    (data_cnt),a
03F3' C2 0407'      jp     nz,new_byte

```


MACRO-80 3.44 09-Dec-81 PAGE 1-14

```

03F6' 3A 011B"      ld      a,(datrd)      ; increment databuffer read pointer
03F9' 3C             inc      a
03FA' E6 0F         and      0fh
03FC' 32 011B"      ld      (datrd),a

03FF' C2 0407'      jp      nz,new_byte    ; signal host computer wrap around
0402' 3E 16         ld      a,SYN          ; has occurred (PC can synchronise)
0404' CD 0000*      call     send##

0407'              new_byte:
0407' 3A 009D"      ld      a,(data_cnt)
040A' B7            or      a
040B' C2 0425'      jp      nz,new_bit

040E' 79            ld      a,c            ; get new byte from data buffer
040F' 07            rlc                     ; and place it in the data table
0410' 07            rlc
0411' 07            rlc
0412' 07            rlc
0413' 5F            ld      e,a
0414' 16 00         ld      d,0
0416' 21 00A4"      ld      hl,datbuf
0419' 19            add     hl,de
041A' 3A 011B"      ld      a,(datrd)
041D' 5F            ld      e,a
041E' 19            add     hl,de
041F' 7E            ld      a,(hl)
0420' 21 0096"      ld      hl,data_tab
0423' 09            add     hl,bc
0424' 77            ld      (hl),a

0425'              new_bit:
0425' 21 0096"      ld      hl,data_tab    ; get new data bit to be modulated
0428' 09            add     hl,bc          ; from data table
0429' 7E            ld      a,(hl)
042A' 07            rlc
042B' 77            ld      (hl),a
042C' DA 043A'      jp      c,negbit      ; zero or one data bit?
042F' 3A 007B"      ld      a,(patt_0)    ; get modulation and shift pattern for 0-bit
0432' 5F            ld      e,a
0433' 3A 007D"      ld      a,(shift_0)
0436' 57            ld      d,a
0437' C3 0442'      jp      1121
043A'              negbit:
043A' 3A 007C"      ld      a,(patt_1)    ; else get modulation and shift pattern for 1-bit
043D' 5F            ld      e,a
043E' 3A 007E"      ld      a,(shift_1)
0441' 57            ld      d,a

0442' 21 0088"      1121: ld      hl,patt_tab ; fill in modulation pattern for databit
0445' 09            add     hl,bc
0446' 73            ld      (hl),e
0447' 21 008F"      ld      hl,shift_tab ; fill in shift pattern for databit
044A' 09            add     hl,bc

```

MACRO-80 3.44 09-Dec-81 PAGE 1-15

```

044B' 72          ld      (hl),d
044C'          nxt_ret:
044C' F3          di          ; the subroutine "prep_next" will (at least)
044D' 3A 009F"    ld      a,(timeleft) ; have taken 424 states, so "timeleft" can
0450' D6 10      sub      16          ; be decreased 16 units of 25 states
0452' 32 009F"    ld      (timeleft),a
0455' FB          ei

```

```

0456' C1          pop      bc
0457' C9          ret

```

```

*****
;*
;* Subroutine : Next_stat
;* Function : Switch to next station in chain.
;* Inputs : bc = index of current station
;* Outputs :
;*          xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;*          xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
;*
;*
*****

```

```

0458'          next_stat:
0458' 2A 0011"    ld      hl,(temprem) ;save temprem for "set_del"
045B' 22 0013"    ld      (temprem2),hl
045E' CD 01CB'    call set_timer ;calculate new quotient, tempquo,
;remainder and temprem for burst after the
;next one
0461' CD 031C'    call set_del ;set delay lines
0464' 3A 0000"    ld      a,(index) ;get index of current station
0467' 4F          ld      c,a
0468' 06 00      ld      b,0
046A' 3E 07      ld      a,7 ;set burstcount to 7
046C' 32 0002"    ld      (brsct),a
046F' 21 0004"    ld      hl,phctab ;invert phase code of current station for
0472' 09          add     hl,bc ;next GRI
0473' 7E          ld      a,(hl)
0474' EE 55      xor      01010101b
0476' 77          ld      (hl),a
0477' 3A 009E"    ld      a,(nxt_index) ;increment station index
047A' 32 0000"    ld      (index),a
047D' 4F          ld      c,a

```


MACRO-80 3.44 09-Dec-81 PAGE 1-16

```

047E' B7          or      a
047F' C2 0486'    jp      nz,116
0482' 3E 00      ld      a,0
0484' D3 82      out      (trig1),a      ;generate trig1 pulse
0486' 21 0004"    116: ld      hl,phctab      ;select new phasecode
0489' 09          add     hl,bc
048A' 7E          ld      a,(hl)
048B' 32 0003"    ld      (phcode),a

048E' 21 005F"    ld      hl,g_ecdtb      ;copy gndwave ECD[i]
0491' 09          add     hl,bc
0492' 7E          ld      a,(hl)
0493' 32 0015"    ld      (g_ecd),a

0496' 21 0066"    ld      hl,s_ecdtb      ;copy skywave ECD[i]
0499' 09          add     hl,bc
049A' 7E          ld      a,(hl)
049B' 32 0016"    ld      (s_ecd),a

049E' 3E 00      ld      a,0      ;generate trig2 pulse
04A0' D3 83      out      (trig2),a

04A2' CD 04A9'    call    set_lev      ;set signal levels

04A5' CD 04F2'    call    carriers_on

04A8' C9          ret

```

```

;*****
;*
;* Subroutine : Set_lev
;* Function : set gndwave & skywave signal
;*           levels for next station
;* Inputs : data in gnd_lev & sky_lev
;*           tables
;*           BC = index for next station
;* Outputs : none
;*
;*****

```

```

04A9'          set_lev:
04A9' 1E 30      ld      e,0110000b      ;40 dB att. control bits

04AB' 21 0051"    ld      hl,gnd_lev      ;send gndwave level to LOGDAC data port
04AE' 09          add     hl,bc
04AF' 7E          ld      a,(hl)
04B0' FE 69      cp      105      ;40 dB att. on or off ?
04B2' 3B 04      jr      c,glev_ok
04B4' D6 69      sub     105
04B6' CB A3      res     4,e
04B8'          glev_ok:
04B8' D3 18      out      (dacdata),a

```

MACRO-B0 3.44 09-Dec-81

PAGE 1-17

```

04BA* 3E 07          ld      a,00000111b      ;enable DAC CS-lines
04BC* D3 1C          out     (daccont),a
04BE* CB 87          res     0,a              ;generate WR-pulse for gndwave DAC
04C0* D3 1C          out     (daccont),a
04C2* CB C7          set     0,a
04C4* D3 1C          out     (daccont),a

04C6* 21 0058*       ld      hl,sky_lev       ;send skywave level to LOGDAC data port
04C9* 09             add     hl,bc
04CA* 7E             ld      a,(hl)
04CB* FE 69          cp      105              ;40 dB att. on or off ?
04CD* 38 04          jr      c,slev_ok
04CF* D6 69          sub     105
04D1* CB AB          res     5,e
04D3*               slev_ok:
04D3* D3 18          out     (dacdata),a

04D5* 3E 05          ld      a,00000101b      ;generate WR-pulse for skywave DAC
04D7* D3 1C          out     (daccont),a
04D9* CB CF          set     1,a
04DB* D3 1C          out     (daccont),a

04DD* 3A 0018*       ld      a,(noise)        ;set noise level
04DE* D3 18          out     (dacdata),a
04E2* 3E 03          ld      a,00000011b      ;generate WR-pulse for noise DAC
04E4* D3 1C          out     (daccont),a
04E6* CB D7          set     2,a
04E8* D3 1C          out     (daccont),a

04EA* CB DF          set     3,a              ;disable DAC CS-lines
04EC* D3 1C          out     (daccont),a

04EE* B3             or      e                ;set 40 dB att. control bits
04EF* D3 1C          out     (daccont),a

04F1* C9             ret

```

```

*****
;*
;*      Subroutine :   Carriers_on
;*      Function   :   Switch carriers on, set phase code & polarity
;*      Inputs    :   phcode = phasecode to use
;*      Outputs   :   Carriers are switched on
;*
*****

```

```

04F2*               carriers_on:
04F2* 21 002B       ld      hl,40              ;wait +/- 200 us to prevent phase
04F5* 2B            del2: dec     hl              ;modulation of the LPF
04F6* 7C            ld      a,h
04F7* B5            or      1
04F8* C2 04F5*     jp      nz,del2
04FB* 3A 0003*     ld      a,(phcode)          ;rotate phase code

```



```

MACRO-80 3.44    09-Dec-81    PAGE    1-18

04FE* 07                      r1ca
04FF* 32 0003*                ld      (phcode),a

;                               r1a                      ;shift phase code to bit 0
0502* C5                      push    bc
0503* 47                      ld      b,a
0504* 3A 0017*                ld      a,(sign)          ;xor phase code bit with sign bit
0507* A8                      xor     b
0508* C1                      pop     bc
0509* CB 47                  bit     0,a
050B* 20 17                  jr      nz,set_b1          ; + or - code ?

050D* 3A 0015*                ld      a,(g_ecd)          ;output gndwave ECD control bits
0510* CB F7                  set     6,a
0512* D3 10                  out     (gndecd),a
0514* CB B7                  res     6,a                ;enable carrier
0516* D3 10                  out     (gndecd),a

0518* 3A 0016*                ld      a,(s_ecd)          ;output skywave ECD control bits
051B* CB F7                  set     6,a
051D* D3 14                  out     (skyecd),a
051F* CB B7                  res     6,a                ;enable carrier
0521* D3 14                  out     (skyecd),a

0523* C9                      ret

0524*                          set_b1:
0524* 3A 0015*                ld      a,(g_ecd)          ;output gndwave ECD control bits
0527* CB EF                  set     5,a                ;set sign bit
0529* CB F7                  set     6,a
052B* D3 10                  out     (gndecd),a
052D* CB B7                  res     6,a                ;enable carrier
052F* D3 10                  out     (gndecd),a

0531* 3A 0016*                ld      a,(s_ecd)          ;output skywave ECD control bits
0534* CB EF                  set     5,a                ;set sign bit
0536* CB F7                  set     6,a
0538* D3 14                  out     (skyecd),a
053A* CB B7                  res     6,a                ;enable carrier
053C* D3 14                  out     (skyecd),a

053E* C9                      ret

```

```

*****
*
* Subroutine :   Chk_mov
* Function :    Process external move request
*              (if any)
* Inputs :      Mov_flg = move request flag
*              Mov_tim = shift time
* Outputs :     DTA's modified
*
*****

```

MACRO-80 3.44 09-Dec-81 PAGE 1-19

```

053F'          chk_mov:
053F' 21 006D'      ld    hl,mov_flg      ;get mov_flg for current station
0542' 09          add    hl,bc
0543' 7E          ld     a,(hl)
0544' 87          or     a
0545' C8          ret     z                ;return if no request

0546' 36 00       ld     (hl),0          ;clear flag

0548' C5          push   bc

0549' 21 0074'     ld     hl,mov_tim     ;E = shift time
054C' 09          add    hl,bc
054D' 5E          ld     e,(hl)

054E' 21 0019'     ld     hl,datab       ;dta(i) = dta(i) - shift time
0551' 09          add    hl,bc
0552' 09          add    hl,bc
0553' 09          add    hl,bc
0554' CD 0586'     call   sgn_ext
0557' 7E          ld     a,(hl)
0558' 93          sub    e
0559' 77          ld     (hl),a
055A' 23          inc    hl
055B' 7E          ld     a,(hl)
055C' 9A          sbc     a,d
055D' 77          ld     (hl),a
055E' 23          inc    hl
055F' 7E          ld     a,(hl)
0560' 99          sbc     a,c
0561' 77          ld     (hl),a

0562' 3A 0000'     ld     a,(index)       ;get prev. index
0565' 3D          dec    a
0566' F2 056D'     jp     p,not_arnd
0569' 3A 0001'     ld     a,(nstat)       ;wrap around if master
056C' 3D          dec    a

056D'          not_arnd:
056D' 4F          ld     c,a
056E' 06 00       ld     b,0
0570' 21 0019'     ld     hl,datab       ;dta (i-1) = dta (i-1) + shift time
0573' 09          add    hl,bc
0574' 09          add    hl,bc
0575' 09          add    hl,bc
0576' CD 0586'     call   sgn_ext
0579' 7E          ld     a,(hl)
057A' 83          add    a,e
057B' 77          ld     (hl),a
057C' 23          inc    hl
057D' 7E          ld     a,(hl)
057E' 8A          adc     a,d
057F' 77          ld     (hl),a
0580' 23          inc    hl
0581' 7E          ld     a,(hl)
0582' 89          adc     a,c

```



```

MACRO-80 3.44 09-Dec-81 PAGE 1-20

0583' 77          ld      (hl),a
0584' C1          pop     bc
0585' C9          ret

0586'          sgn_ext:          ;sign_extend of E to CODE
0586' 0E 00          ld      c,0
0588' 16 00          ld      d,0
058A' 7B          ld      a,e
058B' 17          rla
058C' 00          ret     nc
058D' 0D          dec     c
058E' 15          dec     d
058F' C9          ret

;*****
;*
;*      Subroutine :   Div32
;*      Function :    unsigned 32 bit division
;*      Inputs :      HLDE = 32 bit dividend
;*                   BC = 16 bit divisor
;*      Outputs :     DE = quotient
;*                   HL = remainder
;*
;*****

0590'          div32:
0590' 3E 10          ld      a,16          ;bit count
0592'          divlp:
0592' 29          add     hl,hl          ;HLDE = HLDE * 2
0593' EB          ex      de,hl
0594' 29          add     hl,hl
0595' EB          ex      de,hl
0596' D2 059A'      jp      nc,hl_ok
0599' 23          inc     hl
059A'          hl_ok:

059A' 87          or      a              ;compare BC with 16 MSB's of HLDE
059B' ED 42          sbc     hl,bc
059D' 0A 05A4'      jp      c,not_ok

05A0' 13          inc     de              ;increment quotient
05A1' C3 05A5'      jp      next
05A4'          not_ok:
05A4' 09          add     hl,bc          ;correct HLDE
05A5'          next:
05A5' 3D          dec     a              ;keep count
05A6' C2 0592'      jp      nz,divlp
05A9' C9          ret

```

MACRO-80 3.44 09-Dec-81 PAGE 1-21

;***** data area *****

```

05AA'          dseg

0000"          index:  ds   1          ;station index
0001"          nstat:: ds   1          ;total # stations
0002"          brsent:  ds   1          ;burst counter
0003"          phcode:  ds   1          ;current phase code
0004"          phctab:  ds   7          ;phase code table
0008"          quotient:ds   2          ;dta fraction in 10 us units
000D"          remain:  ds   2          ;dta fraction in 25 ns units
000F"          tempquo: ds   2          ;temporary quotient: contains the 10 us units
                                         ;of dta for one burst. "quotient" contains
                                         ;absolute time, whereas "tempquo" contains
                                         ;eurofix corrected time
0011"          temprem: ds   2          ;temporary remainder. use as "tempquo"
0013"          temprem2:ds   2          ;temporary remainder. Temprem saved to this
                                         ;variable before calculating the new temprem

0015"          g_ecd:   ds   1          ;gndwave ECD control bits
0016"          s_ecd:   ds   1          ;skywave ECD control bits
0017"          sign::   ds   1          ;external controlled sign bit
0018"          noise::  ds   1          ;noise level

0019"          dtatab:: ds  4*7        ;dta data table, 32 bit entries in 25 ns units
0035"          skytab:: ds  4*7        ;skywave delay data table, 16 bit, 25 ns units
0051"          gnd_lev::ds   7          ;gndwave signal levels
0058"          sky_lev::ds   7          ;skywave signal levels
005F"          g_ecdtb::ds   7          ;gndwave ECD data table
0066"          s_ecdtb::ds   7          ;skywave ECD data table

006D"          mov_flg::ds   7          ;move request flags
0074"          mov_tim::ds   7          ;shift time

007B"          patt_0:: ds   1          ;Eurofix pattern for 0 databit
007C"          patt_1:: ds   1          ;Eurofix pattern for 1 databit
007D"          shft_0::  ds   1          ;Eurofix advance/delay shifts for 0 databit

```



```

MACRO-80 3.44 09-Dec-81 PAGE 1-22

007E*      shift_1::ds 1      ;Eurofix advance/delay shifts for 1 databit
007F*      adv_tim::ds 1      ;Eurofix phase advance time
0080*      del_tim::ds 1      ;Eurofix phase delay time
0081*      mod_teb::ds 7      ;Simulator operation mode: 00 = no Eurofix
                                ;                      FF = Eurofix mode
0088*      patt_tab::ds 7
008F*      shift_tab::ds 7
0096*      data_tab::ds 7      ;present data bits
009D*      data_cnt::ds 1      ;how many data bits are used?

009E*      nxt_index::ds 1      ;index of following station in chain

009F*      timeleft::ds 1      ;time left to wait
00A0*      store1::ds 2        ;general word to save information
00A2*      store2::ds 2        ;general word to save information

00A4*      datbuf::ds 7*16      ;buffer for the databits, 16 places for
                                ;each sender
0114*      datwr::ds 7          ;databuffer write pointer
011B*      datrd::ds 1          ;databuffer read pointer

011C* 00      db 0
                                end

```

MACRO-80 3.44 09-Dec-81 PAGE 5

Macros:

CMPHL COMPI6

Symbols:

0208' ..0000	0278' ..0001	02E0' ..0002
0363' ..0003	0127' ADV_INIT	007F1" ADV_TIM
0002" BRSCNT	0197I" BRS_INT	04F2' CARRIERS_ON
053F' CHK_MOV	001C DACCONT	0018 DACDATA
009D" DATA_CNT	0096" DATA_TAB	00A4I" DATBUF
011B1" DATRD	01141" DATWR	04F5' DEL2
03A4' DELAY	000C DELPORT	0128' DEL_INIT
0080I" DEL_TIM	0590I' DIV32	0592" DIVLP
00E4' DTAINIT	0019I" DTATAB	0115' ECDINIT
03C9' FIRST	04B8' GLEV_OK	0004 GNDDEL
0010 GNDECD	0051I" GND_LEV	0015" G_ECD
005F1" G_ECDTB	0107' G_LEVINIT	059A' HL_OK
0000" INDEX	0000I" INITBRS	0080 INTCL1
0081 INTCL2	0223' INTER_STAT	0311' LL10
032D' LL11	0333' LL12	03EF' LL20
0442' LL21	03AA" LL30	03CD' LL5
0486' LL6	0232' LL9	0129' MODE_INIT
0081I" MOD_TAB	006D1" MOV_FLG	011C' MOV_INIT
0074I" MOV_TIM	043A' NEGBIT	02AB' NEGSHFT
0425' NEW_BIT	0407' NEW_BYTE	05A5' NEXT
0109' NEXT_DTA	0458' NEXT_STAT	0018I" NOISE
02F2' NOSHIFT	056D' NOT_ARND	05A4' NOT_OK
0001I" NSTAT	009E" NXT_INDEX	044C" NXT_RET
0078I" PATT_0	007C1" PATT_1	0123' PATT_INIT
0088" PATT_TAB	00DD' PHCINIT	0003" PHCODE
0004" PHCTAB	03BC' PREP_NEXT	000B" QUOTIENT
000D" REMAIN	0212' RM_OK2	0282' RM_OK3
02C4' RM_OK4	036B' RM_OK5	02E8' RM_OK8
02A1' RM_OK9	0405" SEND	0524' SET_B1
031C' SET_DEL	04A9' SET_LEV	01CB' SET_TIMER
0586' SGN_EXT	007D1" SHFT_0	007E1" SHFT_1
0125' SHFT_INIT	008F" SHFT_TAB	0017I" SIGN
0008 SKYDEL	0014 SKYECD	00F9' SKYINIT
0035I" SKYTAB	0058I" SKY_LEV	04D3' SLEV_OK
0130I" START	00A0" STORE1	00A2" STORE2
0016 SYN	0016" S_ECD	0066I" S_ECDTB
010E' S_LEVINIT	000F" TEMPQUO	0011" TEMPREM
0013" TEMPREM2	009F" TIMELEFT	0000 TIMER0
0001 TIMER1	0002 TIMER2	0003 TIMMOD
0082 TRIG1	0083 TRIG2	0084 TRIG3
0085 TRIG4		

No Fatal error(s)

□

Literature references

- [1] R.B. Kellenbach,
Ontwerp en realisatie van een Dynamische Dual-Chain Loran-C Simualtor
Delft University of Technology, Faculty of Electrical Engineering, deperament of
Electronic Technology, The Netherlands, 1987
- [2] Per K. Enge, Rudolph M. Kalafus, Michael F. Ruane,
Differential operation of the Global Positioning System
IEEE Communications Magazine, Vol. 26, No. 7, July 1988
- [3] D. van Willigen,
Eurofix, a Synergism of Navstar/GPS and Loran-C
The Netherlands, Delft University of Technology, Faculty of Electrical Engineering,
Telecommunication and Traffic Control Systems
- [4] D. van Willigen,
Integrated Eurofix and IALA's DGPS: Improved Integrity and Availability
The Netherlands, Delft University of Technology, Faculty of Electrical Engineering,
Telecommunication and Traffic Control Systems
- [5] Peter Norton,
Handboek voor IBM programmeurs
Deventer: Kluwer Technische Boeken B.V. ISBN 90-201-2054-9, 1988
- [6] Hewlett-Packard Corporation
HP Vectra Technical Reference Manual Volume 1: Hardware
Sunnyvale, USA: Hewlett-Packard Corporation, 1985
- [7] Intel
MCS-85 User's Manual
Santa Clara CA, USA: Intel Corporation, 1978
- [8] L.J. Beekhuis,
The Asynchronous Correction Concept for High Precision DGPS in Eurofix,
The Netherlands, 1993, Delft University of Technology, Faculty of Electrical
Engineering, Telecommunication and Traffic Control Systems