Design and implementation of a microprocessor-controlled baseband spread-spectrum system

Diederen, J.H.P.

Award date:
1993

EINDHOVEN UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING
TELECOMMUNICATIONS DIVISION

Design and implementation of a microprocessor-controlled baseband spread-spectrum system

by J.H.P. Diederen

Report of graduation work,
performed from August 1992 to August 1993.

Professor: prof.dr.ir. G. Brussaard.
Supervisors: ir. F.L.A.J. Kamperman,
ir. A.P. Verlijsdonk.

Summary

In recent years, the exploitation of satellite communication systems employing Very Small Aperture Terminals (VSATs) has increased rapidly. The trend is to develop even smaller terminals, which are more practical and user-friendly, but have low bit rates. An extremely small VSAT is called a picoterminal and has an antenna diameter of up to several decimeters. To exploit a satellite communication system, the use of a multiple-access technique is necessary. The multiple-access technique most suitable for a picoterminal communication system is Code Division Multiple Access (CDMA), also called Spread-Spectrum Multiple Access (SSMA). At the Telecommunications Division, a project was started to study the feasibility of picoterminal satellite communication networks using spread-spectrum modulation. One of the objectives of this project is to come to a realization of such a picoterminal.

As part of the picoterminal project, the hardware of a baseband picoterminal transceiver was designed. The hardware is based on the Texas Instruments TMS320C25 digital signal processor, which operates at the maximum clock frequency of 50 MHz. The baseband transceiver is used to implement a system for acquisition-time measurements using the single-dwell serial-search method. With this system, measurements were set up to determine the optimal detection threshold of the acquisition system. With this optimized system, acquisition-time measurements were performed for various signal-to-interference ratios. It is concluded that for a practical system the number of channel users must stay below a certain number to guarantee correct operation of the acquisition system.

Since no theory was available for the acquisition-time characteristics of a baseband spread-spectrum system, the measurement results could not be verified. It is not clear if the BPSK-system theory, described in the literature, may be used for a baseband system. Within the time constraints, it was not possible to check this.

The hardware was designed to implement a complete baseband picoterminal. Until now, only the single-dwell acquisition system was implemented. In future, other acquisition methods can be applied, and, in addition, also code tracking can be implemented. Furthermore, the bit-error rate of the system can be tested.

In the hardware design, many EMC measures were applied to reduce interference caused by components and connections on the printed board designed and to other printed circuit boards of the modem. It was demonstrated that these measures are necessary for correct operation of an electronic circuit using high clock frequencies.

List of symbols and abbreviations

List of symbols

Symbol	Meaning	Unit
B_{info}	bandwidth of the data signal	Hz
B_{tr}	bandwidth of the spreaded data signal	Hz
$c(t)$	spreading signal	
$d(t)$	data signal	
δ_{jM}	Kronecker delta-function	
Δf_c	frequency shift	Hz
$\Delta\omega$	frequency uncertainty in a single cell	Hz
$\Delta\Omega$	frequency uncertainty region	Hz
Δt	code-phase uncertainty in a single cell	s
ΔT	code-phase uncertainty region	s
η	threshold level of a single-dwell system	V
η_i	threshold level of the ith detector of a M-dwell system	V
f_c	chip frequency	Hz
ϕ	phase shift	
G_p	processing gain	
$h(t)$	impulse response	
K	variable indicating the penalty time (single-dwell system)	
K_M	variable that indicates the penalty time (M-dwell system)	
M	maximum number of threshold tests within a cell	
μ	mean code-phase update	
N	number of chips per spreading-sequence period	
ω_0	radian carrier frequency	rad/s
P	power	W
P_D	detecion probability	
$P_{D,i}$	conditional detection probability of the ith detector	
P_{FA}	false-alarm probability	
$P_{FA,i}$	conditional false-alarm probability of the ith detector	
P_{FL}	false lock probability after i threshold exceedings	
q	number of cells in the code-phase uncertainty region	

Q	clock-signal period	s
S/N	signal-to-noise ratio	
$s_t(t)$	spreaded modulated data signal	
σ	variance	
$t_a(A)$	read data access time from address valid	s
$t_a(SL)$	read data access time from *STRB low	s
t_{ACE}	time from *CE low to data valid	s
t_{DOE}	time from *OE low to data valid	s
t_{INV}	delay time of inverter	s
$t_{su}(A)$	address setup time before *STRB low	s
$t_{su}(D)W$	data write setup time before *STRB high	s
T	bit period	s
T_{ACQ}	acquisition time	s
T_c	chip period	s
T_d	transmission delay	s
T_d'	time delay of local spreading sequence	s
T_{FL}	out-of-lock time of tracking system after a false lock	s
T_p	penalty time	s
τ_d	dwell time of a single-dwell system	s
$\tau_{d,i}$	dwell time of the i^{th} integrator of a M-dwell system	s
Z_i	output of the i^{th} integrator	V

List of abbreviations

Abbreviation	**Meaning**
ADC	Analog-to-Digital Converter
ALU	Arithmetic Logic Unit
ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit
bps	bits per second
BPSK	Binary Phase-Shift Keying
CCD	Charge-Coupled Device
CDMA	Code Division Multiple Access
CMOS	Complementary Metal-Oxide Semiconductor
cps	chips per second
CRC	Cyclic Redundancy Check
DAC	Digital-to-Analog Converter

DIP	Dual-In-Package
DS	Direct-Sequence
EMC	Electromagnetic Compatibility
EPROM	Erasable Programmable Read-Only Memory
FH	Frequency-Hop
FM	Frequency Modulation
IC	Integrated Circuit
IRQ	Interrupt Request
I/O	Input/Output
LPT	Line Printer
LSB	Least-Significant Bit
MSB	Most-Significant Bit
OpAmp	Operational Amplifier
PC	Personal Computer
PCB	Printed Circuit Board
pdf	power spectral density
PN	Pseudo-Noise
RAM	Random-Access Memory
SAW	Surface Acoustic Wave
SHA	Sample-Hold Amplifier
SS	Spread-Spectrum
SSMA	Spread-Spectrum Multiple Access
SWDS	Software-Development System
TTL	Transistor-Transistor Logic
VCO	Voltage-Controlled Oscillator
VSAT	Very Small Aperture Terminals

Contents

1 Introduction

In recent years, the exploitation of satellite communication systems employing Very Small Aperture Terminals (VSATs) has increased rapidly. A VSAT is a satellite ground station with an antenna diameter of approximately 1 to 2½ meters. The trend is to develop even smaller terminals, which are more practical and user-friendly, but have low bit rates. An extremely small VSAT is called a picoterminal and has an antenna diameter of up to several decimeters. It can operate at data rates of several tens to a few hundred bits per second. Possible applications of a picoterminal satellite communication system are for example data-collection platforms and alarm systems, consisting of a number of distant terminals and one central hub-station, that collects the data. Such a system is illustrated in Figure 1.1.

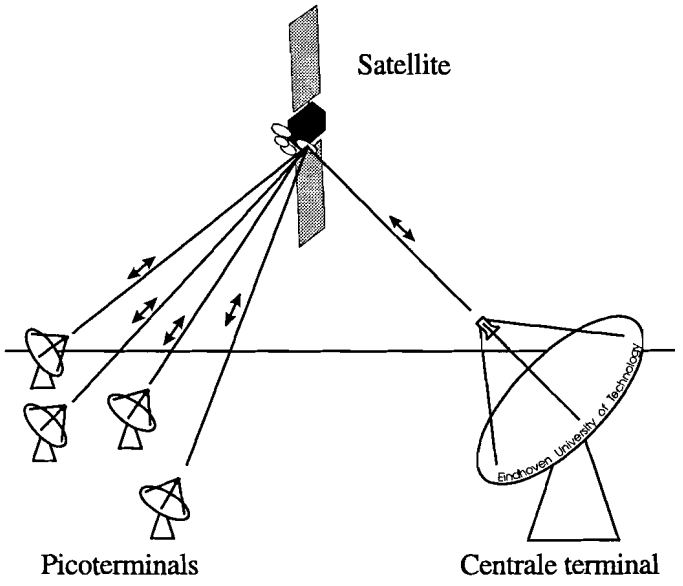


Figure 1.1: A picoterminal satellite communication system.

To exploit a satellite communication system, the use of a multiple-access technique is necessary. This technique allows a number of users to communicate via a common satellite transponder. There are a few possible multiple-access techniques which are commonly applied. Frequency-division multiple access (FDMA) is a technique, whereby several earth stations transmit simultaneously, but on different preassigned frequencies towards a transponder. In time-division multiple access (TDMA), each user is allocated a time slot in which information can be transmitted. Finally, in code-division multiple access (CDMA), also called spread-spectrum multiple access (SSMA), the signal of each user is modulated with a unique spreading signal. The resulting wideband signals from all users

may use the same channel at the same time. CDMA is most suitable for a picoterminal communication system operating at low data rates. It is a simple technique that does not need any network synchronization or management.

At the Telecommunications Division, a project was started to study the feasibility of picoterminal satellite communication networks using spread-spectrum modulation. One of the objectives of this project is to come to a realization of such a picoterminal. The work presented in this report describes the design and implementation of a baseband picoterminal transceiver based on a digital signal processor.

Chapter 2 discusses spread-spectrum communications and two basic techniques are explained. These are direct-sequence spread spectrum and frequency-hop spread spectrum. It is explained why direct-sequence spread spectrum is used in a picoterminal communication system. In Chapter 3, a number of methods for initial synchronization, also called acquisition, in direct-sequence spread-spectrum systems is discussed. These are serial-search acquisition and acquisition using matched filtering. The single-dwell serial-search acquisition method is implemented on the baseband picoterminal transceiver. The transceiver is used to build a measurement system to perform acquisition-time measurements for different numbers of other channel users. The acquisition time characterizes the performance of the acquisition system. With the measurement system, the influence of other channel users on the performance of the acquisition system can be measured. The implementation of the acquisition method is discussed in Chapter 4. Chapter 5 describes the picoterminal-transceiver hardware and in Chapter 6, a communications interface is presented, which provides communication between a baseband picoterminal transceiver and a personal computer. Chapter 7 discusses the software developed for the implementation of the acquisition method. Chapter 8 explains the measurement system and measurement results are presented. Finally, Chapter 9 discusses conclusions and recommendations.

2 Spread-spectrum communications

2.1 Introduction

This chapter gives a general introduction to spread-spectrum communications. Section 2.3 gives a short introduction to maximal-length sequences. Two basic spread-spectrum techniques are discussed in Sections 2.4 and 2.5, and in addition, it is explained why direct-sequence spread-spectrum is used for picoterminal satellite communication systems. Furthermore, synchronization in direct-sequence spread-spectrum systems is discussed.

2.2 Spread-spectrum communications

Spread-spectrum communications were initially used for military applications but in recent years the civil use of these techniques has developed rapidly. The term "spread spectrum" refers to the modulation technique used. It is a modulation technique applied to digital communications and it has the following characteristics:

1) the bandwidth of a modulated signal is much larger than the bandwidth of the information signal;
2) this larger bandwidth is defined by a so called spreading signal, and is independent of the information signal.

Some other modulation techniques also have the first characteristic. For example, frequency-modulated (FM) signals can have a bandwidth that is larger than the bandwidth of the information signal. However, the bandwidth of a FM signal is defined by the modulation index and by the information signal, and thus, frequency modulation does not have the second characteristic. One only speaks of spread-spectrum modulation if it has both properties.

Spread spectrum is used for three major reasons. Firstly, the modulated signals have a low spectral density which implies a low probability of detection. This property is used in military communications for signal hiding. For civil communications it has little use. It only implies that a spread-spectrum system does not interfere much on other systems. Secondly, a spread-spectrum system has a high level of interference rejection. In military communications this is important, because the enemy is not able to deregulate

communications. This property also enables spread spectrum to be applied as a multiple-access technique. The users of the system use the same channel to communicate and each user can reject the interference of the other users up to a certain level. Morover, no user protocol is necessary when applying SSMA, since each user can start communications independent from the other users. In civil communications, this is the main reason why spread-spectrum modulation is applied. Finally should be mentioned that spread-spectrum communication is very robust against jamming and multipath propagation.

The amount of performance improvement that is achieved through the use of spread spectrum techniques is defined as the processing gain G_p of the system. It is defined as the ratio between the signal-to-noise ratio at the output of the receiver, and the signal-to-noise ratio at the input. In practice, this definition can be reduced to the ratio between the bandwidth of the modulated signal B_{tr} and the information-signal bandwidth B_{info}.

$$G_p = \frac{(S/N)_{out}}{(S/N)_{in}} \approx \frac{B_{tr}}{B_{info}} \; . \tag{2.1}$$

Two basic spread-spectrum techniques are direct-sequence spread spectrum and frequency-hop spread spectrum. The principles of these techniques are the subject of Sections 2.4 and 2.5. The next section discusses maximal-length sequences. These sequences are often used as the spreading signal and are also used to generate other spreading codes like Gold codes.

2.3 Maximal-length sequences

2.3.1 Introduction

Data modulation or spreading in spread-spectrum systems is performed by modulating a datasignal with a spreading signal. Demodulation is accomplished by correlation of the received signal with a replica of the spreading signal that is synchronised to the received signal. Spread-spectrum demodulation is usually called despreading.

The spreading signal *c(t)* used to spread and despread the data-modulated carrier is usually generated using a shift register whose contents during each time interval T_c is some linear or nonlinear combination of the contents of the register in the preceding interval. T_c represents the so called chip time. For a spread-spectrum system to operate efficiently, the spreading signal is selected to have certain properties. For example, the phase of the received signal must be initially synchronized and then tracked by the receiver. These functions are facilitated by choosing *c(t)* to have a two-valued autocorrelation function, as

exhibited by the maximal-length sequences considered in the next paragraph. For the receiver it is easy to check whether it is synchronized to the received signal by monitoring the autocorrelation determined after despreading. When the spread-spectrum system is used for multiple access, sets of spreading signals must be found which have good cross-correlation properties. The reason for this is that, the interference of other users after despreading, i.e. the cross-correlation, is kept to a minimum, which results in a better performance. Furthermore, it is often desirable to employ a spreading signal having a very wide bandwidth. This implies that simple spreading code generators which can operate at high speeds should be considered. The earlier mentioned maximal-length sequences or m-sequences, have two-valued autocorrelation, have reasonable cross-correlation properties as compared with other sequences, and are easy to generate. Therefore, m-sequences are used as the initial spreading signal in a picoterminal communication system. In future, also other sequences, like Gold codes can be considered.

2.3.2 Properties of maximal-length sequences

This section discusses the properties and generation of maximal-length sequences. A more detailed discussion can be found in e.g. [2].

M-sequences are generated by a linear-feedback shift register as illustrated in Figure 2.1. A linear-feedback shift register consists of a cascade of single-stage shift registers, modulo-2 adders, and modulo-2 multipliers. The only condition to generate m-sequences is that index g_r=1 and that the polynomial defined by the indices g_i is primitive [2].

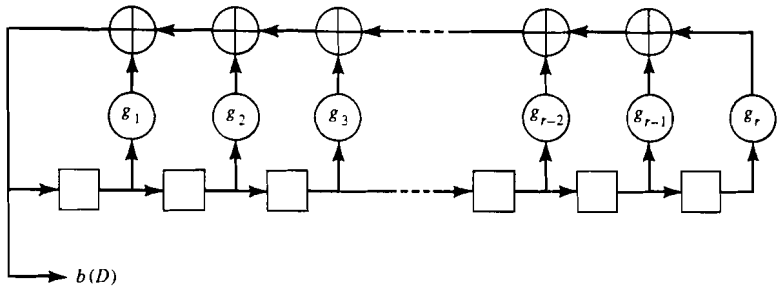


Figure 2.1: A linear-feedback shift register [2].

Maximal-length sequences have a number of properties which are useful in their application to spread-spectrum systems. The most important ones are listed below.

- m-sequences are periodical and have a length of $N=2^r-1$, where r is the length of the generating shift register;
- a maximal-length sequence contains one more one than zero; the number of ones in the sequence is $\frac{1}{2}(N+1)$;

- the output of the shift register can be used to generate a bipolar spreading signal $c(t)=\pm 1$, which is applied in direct-sequence spread spectrum discussed in the next section. The periodic discrete autocorrelation function $R_c(k)$ of $c(t)$ is two-valued and is given by

$$R_c(k) = \begin{cases} 1 & k=l\cdot N \\ -\dfrac{1}{N} & k\neq l\cdot N \end{cases}, \tag{2.2}$$

where l is any integer and N is the sequence period. Figure 2.2 shows the continuous autocorrelation function of a m-sequence.

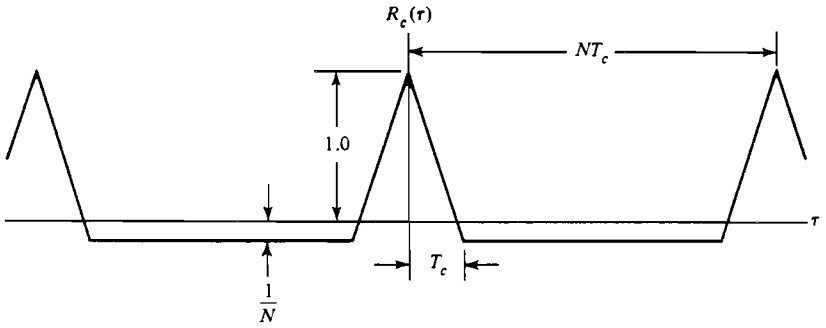


Figure 2.2: Autocorrelation of a maximum-length sequence [2].

From the continuous autocorrelation function, the power spectral density, which is the Fourier-transform of the autocorrelation function, can be determined. It is given by [2]

$$S_c(f) = \sum_{m=-\infty}^{\infty} P_m \delta(f-mf_0), \tag{2.3}$$

where $P_0 = \dfrac{1}{N^2}$, $P_m = \left[\dfrac{N+1}{N^2}\right] sinc^2\left(\dfrac{m}{N}\right)$, and $f_0 = \dfrac{1}{NT_c}$. The power spectral density is illustrated in Figure 2.3. It consists of discrete spectral lines at all harmonics of $1/NT_c$. The envelope of the amplitude of these lines is given by P_m except for the dc term that has an amplitude of $1/N^2$.

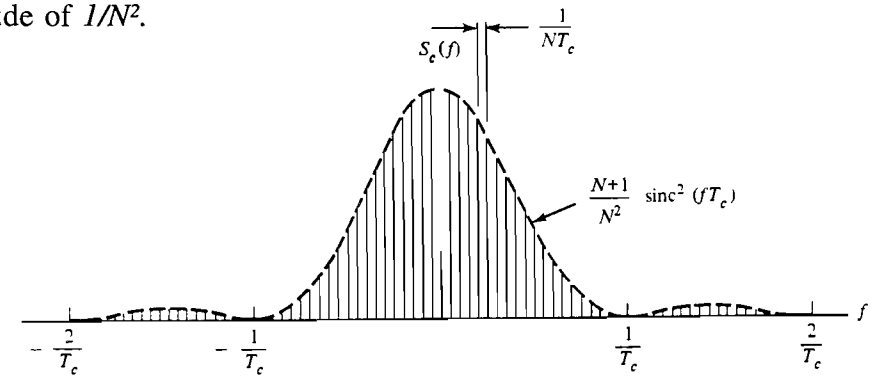


Figure 2.3: The power spectral density of a m-sequence [2].

2.4 Direct-sequence spread spectrum

2.4.1 Introduction

Bandwidth spreading by direct modulation of a data signal with a wideband spreading signal or code is called direct-sequence (DS) spread spectrum. This section discusses direct-sequence spread-spectrum modulation. It explains the principles and characteristics of this modulation technique. The simplest form of direct-sequence spread spectrum employs binary phase-shift keying (BPSK) and spreading modulation. Only BPSK direct-sequence spread spectrum is discussed in this section. A more extensive discussion about direct-sequence spread spectrum in combination with other modulation techniques can be found in literature, e.g. [2].

2.4.2 Principles of direct-sequence spread spectrum

Figure 2.4 illustrates a BPSK direct-sequence spread-spectrum transmitter and receiver. At the transmitter, the data signal *d(t)* is BPSK modulated with a carrier *s(t)* with power *P*. The data signal is characterized by the bit period *T*, or by the bit rate which is *1/T*. Spreading of this signal is accomplished by simply multiplying the BPSK modulated data signal by a function *c(t)*, representing the spreading code. The spreading code is characterized by its bit period T_c, mostly called chip period, or by the chip rate $1/T_c$. The transmitted signal is

$$s_t(t) = \sqrt{2P}\, d(t)c(t)\cos[\omega_0 t] \ . \tag{2.4}$$

Suppose this signal is transmitted via a distorsionless path having transmission delay T_d. The transmitted signal is received together with some type of interference and/or Gaussian noise. At the receiver, demodulation is accomplished by remodulation with the spreading code appropriately delayed to count for the transmission delay T_d, i.e., the received signal is multiplied with a locally generated spreading code $c(t-T_d')$, with $T_d' \approx T_d$. This remodulation or correlation of the received signal with the delayed spreading waveform is called despreading. The despreaded signal is

$$\sqrt{2P}\, c(t-T_d)c(t-T_d')d(t)\cos[\omega_0 t+\phi] \ . \tag{2.5}$$

Since $c(t)=\pm 1$, the product $c(t-T_d)c(t-T_d')$ will be unity if $T_d=T_d'$, that is, if the spreading code is synchronized to the received signal. When correctly synchronized, the signal component of the receiver despreading mixer output is equal to

$$\sqrt{2P}\, d(t)\cos[\omega_0 t+\phi] \ . \tag{2.6}$$

This signal can be demodulated and the data signal *d(t)* can be estimated.

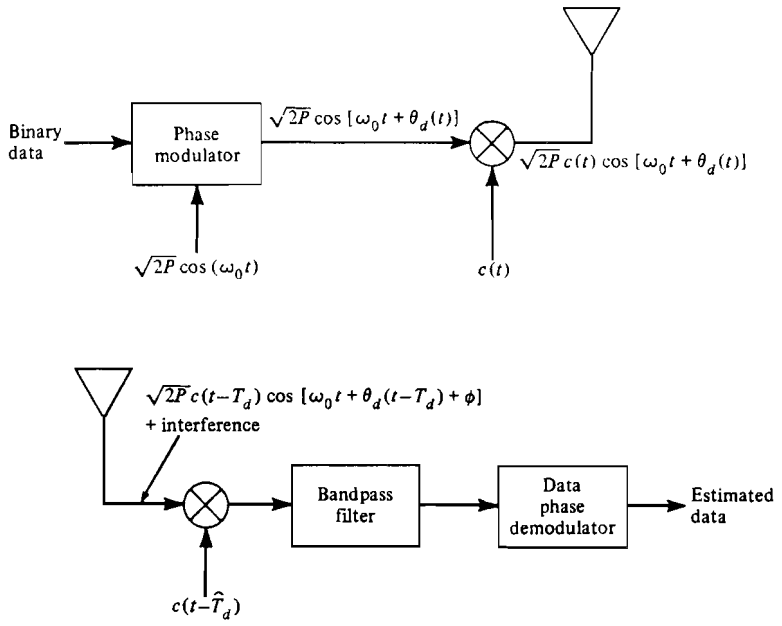


Figure 2.4: BPSK direct-sequence spread-spectrum modulator and demodulator [2].

Figure 2.5 illustrates bandwidth spreading and despreading of a BPSK direct-sequence signal. The data and spreading waveforms are illustrated in Figure 2.5 a) and b). The BPSK modulated data signal is shown in Figure 2.5 c) and the spreaded BPSK modulated data signal in Figure 2.5 d). Figure 2.5 e) shows an incorrectly phased local spreading code, assuming zero propagation delay. Figure 2.5 f) shows the output of the despreading mixer when the local spreading code is incorrectly phased. This signal is not equal to the signal shown in Figure 2.5 c), illustrating that the local spreading code must be synchronized to the received signal. Finally, Figure 2.5 g) shows the despreading mixer output when the spreading code is synchronized to the received signal. The output signal is equal to the BPSK modulated data signal.

As can be concluded from the first three signals shown in Figure 2.5, BPSK modulation of the data signal followed by spreading this signal is the same as spreading the data signal followed by BPSK modulation of this signal. The despreading process shows that despreading of the received signal followed by BPSK demodulation is the same as despreading the signal after BPSK demodulation.

In practice, the spreading signal is periodical and its period is often chosen equal to the bit period of the data signal. This is, however, not necessary for the correct operation of a spread-spectrum system. In the rest of this report it will be assumed that the period of the spreading code, the so called code-length period, is equal to the bit period of the data signal.

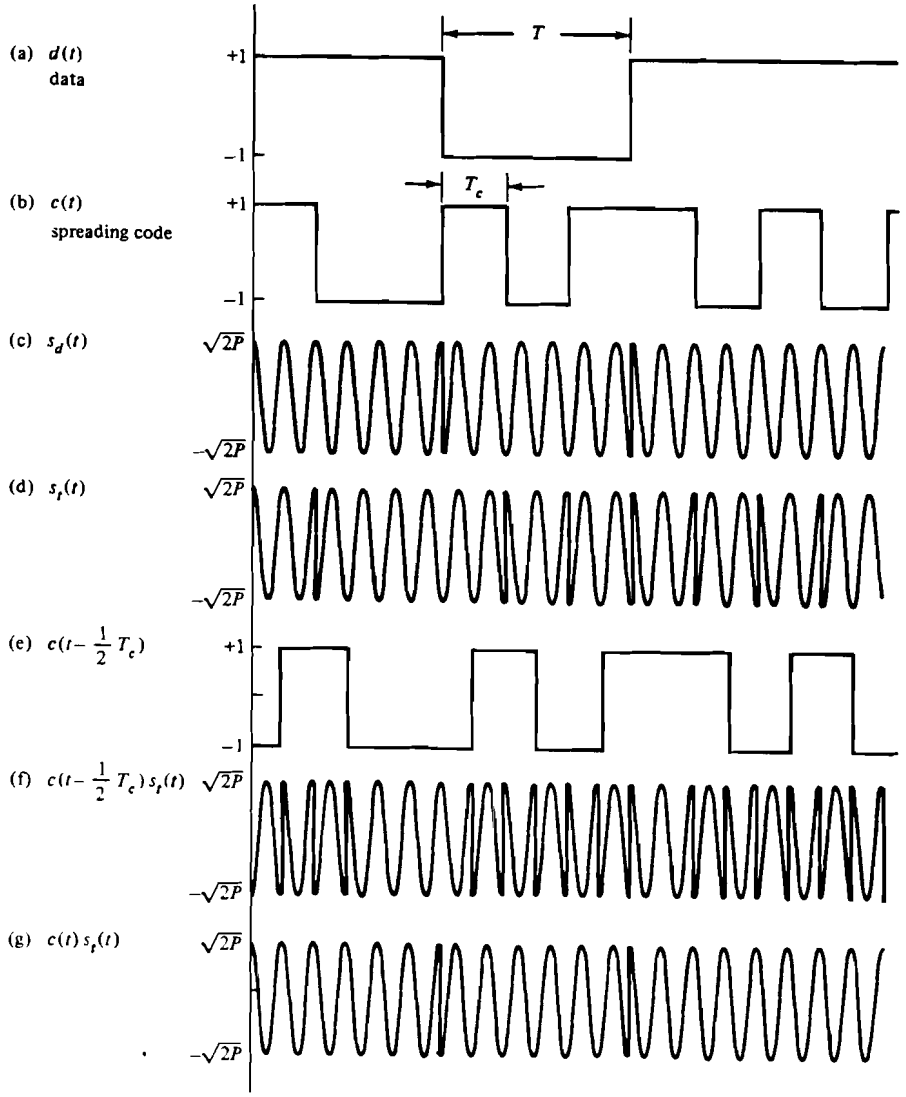


Figure 2.5: BPSK direct-sequence spreading and despreading [2].

The power spectra of the BPSK-modulated and spreaded BPSK-modulated signals are shown in Figures 2.6 and 2.7. Observe that the effect of the modulation by the spreading code is that the bandwidth of the transmitted signal is spreaded by a factor T/T_c, and that this spreading operation reduces the power spectral density with a factor T/T_c. In practical systems, the spreading factor is typically much larger than depicted here.

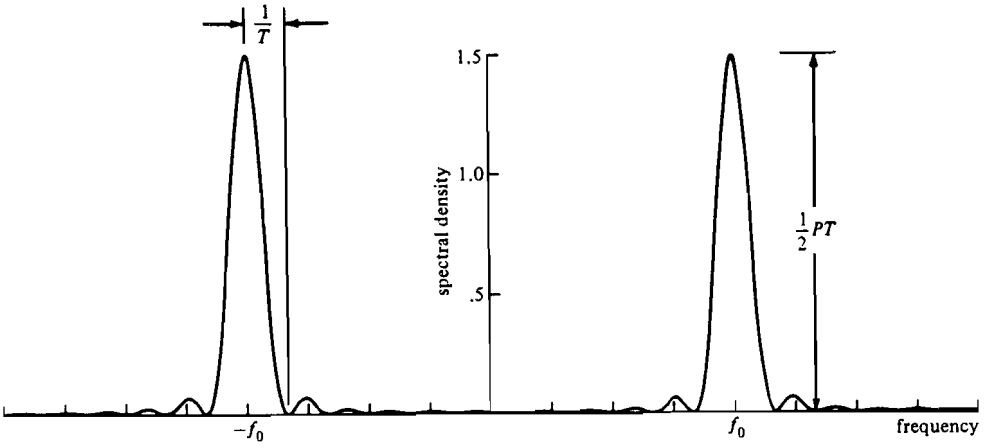


Figure 2.6: Power spectral density of data-modulated carrier [2].

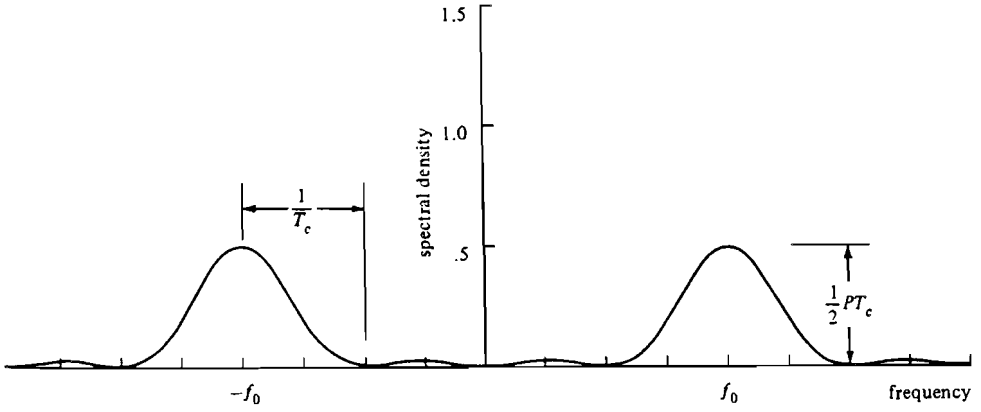


Figure 2.7: Power spectral density of data- and spreading code-modulated carrier [2].

The processing gain G_p of a direct-sequence spread-spectrum system can be reduced to the ratio between the bit rate and the chip rate:

$$G_p \approx \frac{T}{T_c} . \tag{2.7}$$

This ratio was earlier referred to as the spreading factor.

2.5 Frequency-hop spread spectrum

A second method for widening the spectrum of a data-modulated carrier is to change the frequency of the carrier over a set of discrete frequencies. Each carrier frequency is chosen from a set of frequencies, which are spaced approximately the width of the spectrum of the modulated data signal. The spreading code in this case does not directly modulate the data-modulated carrier, but is instead used to control the sequence of carrier

frequencies. Because the transmitted signal appears as a data-modulated carrier which is hopping from one frequency to the next, this type of spread spectrum is called frequency-hop (FH) spread spectrum. In the receiver, the frequency hopping is removed by down-conversion of the received signal with a local oscillator which is hopping synchronously with the received signal.

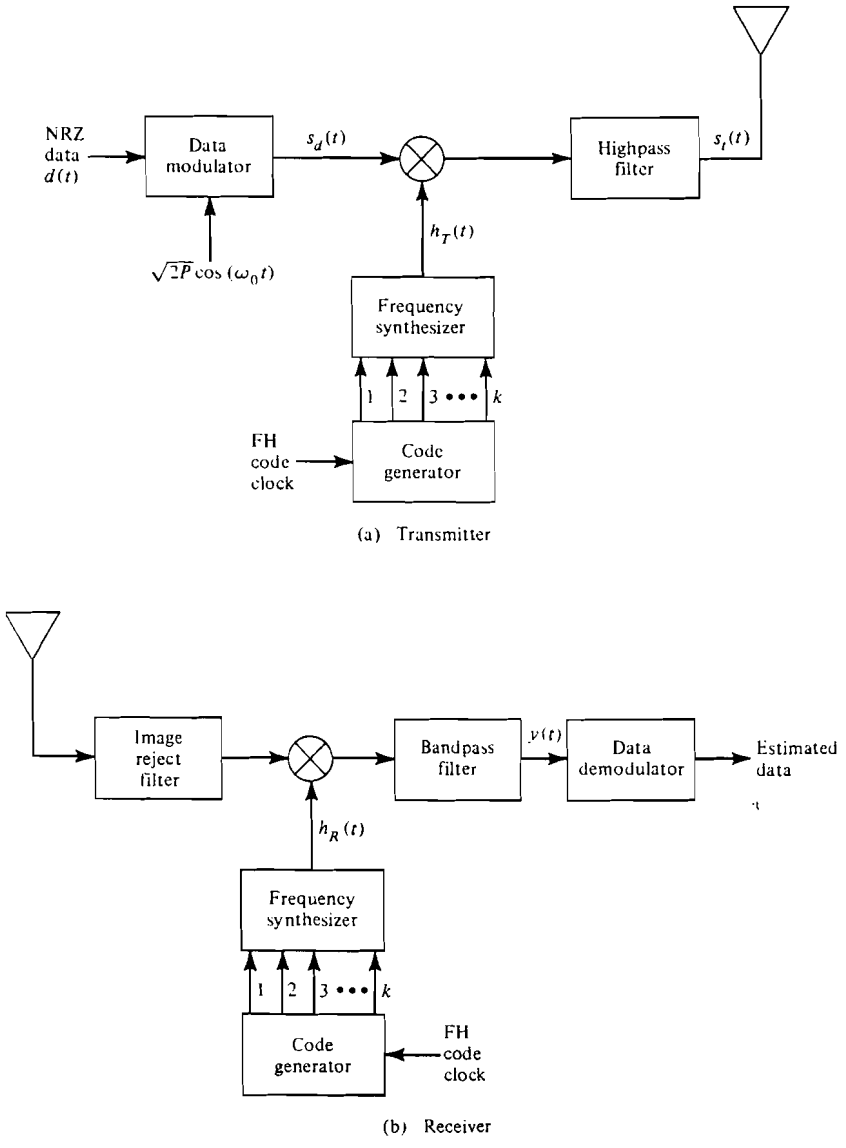


Figure 2.8: Frequency-hop transmitter and receiver [2].

Figure 2.8 shows an example of a frequency-hop spread-spectrum transmitter and receiver. At the transmitter, the data-modulated signal is up-converted with the frequency-synthesizer output signal. The output-signal frequency is clocked each chip period T_c. In contrast to the direct-sequence system, where the spreading code is used one bit at a time, the spreading code here is used k bits at a time. Thus, each carrier frequency is chosen from a set of 2^k frequencies. The transmitted signal is received with some type of interference and/or noise. The received signal is down-converted with the frequency-synthesizer output signal, that is also controlled by the spreading code and synchronized to the received signal.

Frequency-hop spread spectrum is applied in a variety of implementations, such as coherent, non-coherent, and differentially coherent fast or slow frequency-hop systems. It goes beyond the aim of the section to describe all possible implementations of frequency-hop spread spectrum. These techniques are discussed in detail in literature, such as in [2] and [5].

Whether direct-sequence or frequency-hop spread spectrum is applied in a picoterminal satellite communications network is discussed in the next section.

2.6 Spread spectrum in satellite communication networks

Of the two methods discussed in the previous sections, direct-sequence spread spectrum is mostly applied in satellite communication networks because of following three reasons.

Direct-sequence spread spectrum is a technique very suitable in combination with BPSK modulation. It is simple and has good performance. Moreover, the performance of a direct-sequence spread-spectrum system is little better than the performance of a frequency-hop system.

The second reason is that the frequency-hop transmitter and receiver need a frequency synthesizer. A frequency synthesizer makes the frequency-hop spread-spectrum system more expensive than the direct-sequence spread-spectrum system. Therefore, if direct-sequence spread-spectrum can be applied it is the cheapest technique to implement.

Direct-sequence spread spectrum has decreasing performance relative to frequency-hop spread spectrum when the so called "near-far problem" exists. The near-far problem implies that a more adjacent transmitter dominates the signal of a distant transmitter. For a direct-sequence system this implies that the performance of the system decreases rapidly,

since interference can only be rejected up to a certain level. For a frequency-hop system it implies that only few bits are damaged. These bits can be restored using forward error-correction techniques and interleaving which is not further discussed. The near-far problem does not exist for satellite communication systems, because the lengths of the communication paths are approximately equal. This provides us to apply direct-sequence spread spectrum for satellite communication systems.

2.7 Synchronization in direct-sequence spread-spectrum systems

The two basic functions of a direct-sequence spread-spectrum receiver are to despread the received signal, i.e. removing the spreading code, and to regain the information signal. Despreading of the received signal is accomplished by generating a local replica of the spreading code and then synchronising this local spreading code to the input signal. Multiplication or remodulation of the received signal by the synchronized local PN code performs the desired despreading process.

The synchronization process can be divided into two steps. First, initial synchronization or acquisition is performed. The local spreading code is synchronized to the received signal to within a certain code phase, or, in other words, the correct code phase is acquired. This is accomplished by detection of the autocorrelation peak of the maximal-length sequence. When the receiver is synchronized, the local code sequence must be synchronized accurately and track the received signal. This is the second step in synchronization and is referred to as tracking.

Synchronization is very important as was illustrated in Figure 2.5. If there is a small phase difference between the received signal and the local spreading code, the effective processing gain of the system decreases, i.e., not the whole processing gain G_p is used. Therefore the received signal must be tracked accurately to the spreading code.

Chapter 3 discusses initial synchronization or acquisition of the locally generated spreading code to the received signal. Two basic acquisition methods are described and the performance of these methods is discussed. Tracking of the local spreading code is not discussed in this report. However, an extensive description about tracking methods can be found in literature, e.g. [2] and [5].

3 Acquisition

3.1 Introduction

This chapter describes the most common acquisition methods. These are serial-search acquisition and acquisition using matched filtering. For each method, expressions are given for the mean and variance of the acquisition time. In the next chapter, the implementation of the acquisition method is discussed.

Initial synchronization or acquisition of the incoming spreaded waveform is a significant problem in spread-spectrum system design. The locally generated replica of the input PN code must be synchronized in phase as well as in frequency. The correct code phase and frequency must be found quickly. The performance of the acquisition system is characterized by the time to acquire the right phase and frequency, usually denoted as the acquisition time T_{ACQ}, and by the lowest possible signal-to-noise or signal-to-interference ratio for which the system works correctly. The system to be designed will be a compromise between system performance and complexity.

3.2 Serial-search acquisition

3.2.1 Introduction

As stated in the introduction, the local PN code must be synchronized in phase as well as in frequency. It will be assumed that the phase uncertainty between the input signal and the receiver-generated replica as mentioned in the introduction, is within a range ΔT and the frequency uncertainty is within a range $\Delta\Omega$. The phase/frequency uncertainty region may be illustrated as a rectangle with dimensions $\Delta T * \Delta\Omega$, as in Figure 3.1. This rectangle can be subdivided into smaller rectangles, whose dimensions Δt and $\Delta\omega$ are chosen in such a way, that the correct phase/frequency cell can be determined by a single test in each cell of Figure 3.1.

The serial-search method, introduced by Sage [1], implies stepping from cell to cell in the phase/frequency uncertainty region and making a test for synchronization within each cell. There are several methods to make the test for synchronization. These are described in the next sections.

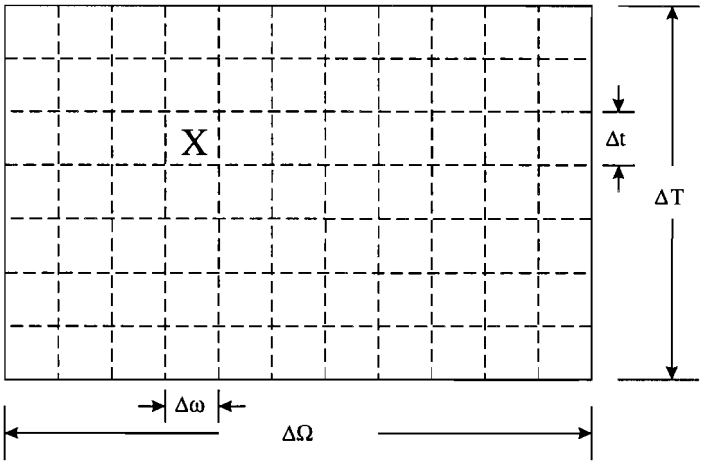


Figure 3.1: The phase/frequency uncertainty region [2].

A phenomenon in satellite communications is a certain frequency shift of the received signal due to Doppler shifts and oscillator instabilities. In first instance, the analysis of serial-search acquisition methods considers that no Doppler shifts and oscillator instabilities exist, thus the frequency uncertainty is zero. After this analysis has been made, the results are evaluated for the case when frequency uncertainty is present. At baseband level, these frequency shifts are very small because the bit rate is very low.

The phase uncertainty region consists of q cells. The magnitude of q depends on the code length N and on the performance of the tracking system. It is common practice to require that the received and local PN codes are aligned to each other within half a code-chip period before the system hands over from acquisition to tracking. According to this requirement, the phase delay of the local PN code must be advanced or retarded in discrete steps of half a chip period, i.e., the phase update is half a chip period. From the above it can be concluded that the number of cells to be searched is $q=2N$.

When a-priori information about the code phase is present, a non-uniform search can be carried out starting in the region (a couple of cells) with the highest code phase probability, and expanding as a function of time to cells of lower probability. In literature, these techniques are referred to as expanding window or z-search strategies. The system to be designed is an asynchronous satellite communication system and no a priori information about the code phase is present, i.e., all cells have equal probability of having the right code phase. Thus, a uniform search from one end of the uncertainty region to the other end is carried out. However, if synchronization was found but lost again, some information about the code phase is present and a z-search strategy can be performed.

In the next two sections, the two most common serial-search methods are described. These are the single-dwell and the multiple-dwell serial search. For each method, the mean and variance of the acquisition time are given.

3.2.2 Single-dwell serial-search acquisition

The simplest form of a serial-search acquisition system is the single-dwell acquisition system, where a single detector is used to examine each of the possible cells for a fixed time until the correct code phase has been located. Figure 3.2 illustrates the baseband version of the single-dwell serial-search acquisition system.

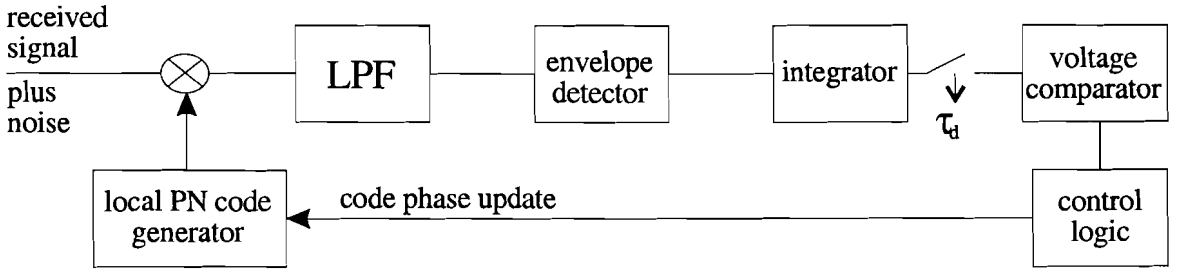


Figure 3.2: Schematic diagram of the single-dwell serial-search acquisition system.

After the received signal is multiplied with a local replica of the PN code, it is passed through a low-pass filter. The filter bandwidth is chosen in such a way that the despreaded signal is passed but the majority of the noise and interference is filtered out. The filter output is then envelope detected to remove the influence of the information signal on the output. The detector output is integrated over a fixed time τ_d in an integrate-and-dump circuit, and then compared to a pre-set threshold. Thus, the integrate-and-dump output is the correlation of the input signal and the local replica of the PN code over a time τ_d. The fixed time τ_d is called the dwell time. In practical systems, the dwell time τ_d is a system parameter which can take values from several chip periods to a complete spreading code period NT_c.

If the integrate-and-dump output is above the threshold level, then a hit is declared. Normally, the system will go into a verification mode before tracking is started. The code phase is verified for one or several dwell times. If a true hit is declared, that is, the correct code phase has been determined, the search ends and tracking of the signal starts. If the hit is a false alarm, that is, the threshold is exceeded but the code phase is not correct, the system steps to the next cell and acquisition continues. The time interval between starting the verification mode and continuing acquisition is called the penalty time T_p, and can be characterized by an extended dwell time $K\tau_d$ ($K \gg 1$). If the integrate-and-dump output falls below the pre-set threshold, the local PN code generator steps to the next cell and the

search proceeds. Of course, synchronization can occur only once per search through the phase uncertainty region, unlike a false alarm , which can occur many times per search.

The probability that a true hit occurs is called the detection probability P_D. The time needed to detect the correct code phase is called the acquisition time T_{ACQ}. The probability that a false alarm occurs is called the false alarm probability P_{FA}. P_D and P_{FA} are dependent on system parameters, like processing gain, dwell time, and threshold setting, and on-link parameters such as signal-to-noise ratio and signal-to-interference ratio.

3.2.2.1 Single-dwell acquisition-time performance

The performance of the acquisition system can be characterized by the time to acquire the right code phase, i.e., the acquisition time T_{ACQ}. This is a random variable dependent on the initial code phase position of the receiver-generated PN code, the detection and false alarm probabilities (P_D and P_{FA}, respectively), the dwell time τ_d, and the penalty time T_p when a false alarm has occurred. The complete statistical description of the acquisition time is the probability density function, but normally it suffices to characterize it by its mean and variance.

The mean and variance of the acquisition time are calculated using the Markov Chain acquisition model. A description of this model can be found in e.g. [3] or [5]. The analysis to calculate the mean and variance of the acquisition time for the single-dwell system is described in [5].

The mean acquisition time for the single-dwell serial-search acquisition system is

$$T_{ACQ,\ mean} = \frac{(2-P_D)(1+KP_{FA})}{2P_D} \cdot q\tau_d \tag{3.1}$$

for $q \gg 1$, which is the case for all systems in practice. The variance of the acquisition time is

$$\sigma^2_{T_{ACQ}} = \tau_d^2(1+KP_{FA})^2 q^2 \left(\frac{1}{12} + \frac{1}{P_D^2} - \frac{1}{P_D} \right). \tag{3.2}$$

As can be seen in Equations (3.1) and (3.2), the mean and variance of the acquisition time are dependent on the false alarm probability P_{FA} and the detection probability P_D. As stated before, these probabilities are dependent on system parameters, such as processing gain, dwell time, and threshold setting, and on link parameters, such as signal-to-noise ratio and signal-to-interference ratio. To characterize P_D and P_{FA} as a function of these parameters, it is necessary to know all signal and noise characteristics at the input of the

acquisition circuit. In literature, an analysis of P_D and P_{FA} as a function of the signal-to-noise ratio at the input of the acquisition system, is only given for BPSK-modulated PN signals, e.g [5]. Within the time constraints, it has not been possible to derive a relation between the acquisition time performance of a BPSK spread-spectrum system and the performance of the baseband system implemented. However, to gain a clear insight in the acquisition-time performance of the single-dwell serial-search method, some practical values are needed. Therefore, a practical example of the acquisition-time performance is given here. Practical values for the detection probability P_D are between 0.7 and 1.0. Usually, the false-alarm probability P_{FA} is less than 0.1.

Example: Consider a synchronous single-dwell acquisition system with a dwell time $\tau_d=NT_c$, that is, the dwell time is equal to the code-length period and $N=127$. Here, T_c is the chip period equal to $T_c=1/f_c$, with f_c the chip frequency equal to 16 kcps. For a synchronous acquisition system only one sample per chip is required. This means that the number of cells to be searched is $q=N$. The detection probability is assumed to be $P_D=0.95$ and the false-alarm probability is $P_{FA}=0.01$. The parameter that characterizes the penalty time is K=3. The mean acquisition time of this system evaluates to

$$T_{ACQ,\,mean} = 0.574 \text{ s},$$

and the variance of the acquisition time becomes

$$\sigma^2_{T_{ACQ}} = 0.150 \text{ s}^2. \tag{3.4}$$

The values for mean and variance of acquisition time calculated in this section characterize the performance of the acquisition system in the absence of frequency shifts. It is a considerable problem to account for the influence of frequency shifts on the acquisition-time performance. One cannot account for frequency shifts exactly, because this is too difficult. Therefore, an approximation is made for the performance of the system in the presence of frequency uncertainty. The frequency shift causes the phase difference between the modulated input signal and the locally generated code sequence to be time varying, i.e. it causes a change in the phase update of the correlator. In literature, e.g. [5], an estimation of this effect on the acquisition time is modelled as follows. The mean phase update is given by

$$\mu = \frac{N}{q} + \Delta f_c \tau_d + \Delta f_c K \tau_d P_{FA}, \tag{3.3}$$

where N/q is the phase update in the absence of frequency uncertainty, $\Delta f_c\tau_d$ is the code phase shift due to frequency shift during the dwell time, and $\Delta f_c K\tau_d$ is the code phase shift during the verification process. The mean and variance of the acquisition time can be calculated by filling in $q=N/\mu$ in Equations (3.1) and (3.2). This results in

$$T_{ACQ,\,mean} = \frac{T_{ACQ,\,mean}\,\Big|_{no\ frequency\ uncertainty}}{1+\frac{q}{N}\Delta f_c \tau_d (1+KP_{FA})} \tag{3.4}$$

$$\sigma^2_{T_{ACQ}} = \frac{\sigma^2_{T_{ACQ}}\,\Big|_{no\ frequency\ uncertainty}}{\left[1+\frac{q}{N}\Delta f_c \tau_d (1+KP_{FA})\right]^2} \tag{3.5}$$

Again it is emphasized that these expressions are just estimates, because the effects of frequency shifts cannot be calculated exactly.

3.2.3 Multiple-dwell serial-search acquisition

The multiple-dwell serial-search technique is an extension of the single-dwell serial-search technique. This technique implies threshold testing by the detector after integration over various time intervals. This variation of integration time is achieved by allowing the examination interval to consist of a series of short fixed dwell periods, each longer than its predecessor, with a decision being made after each. In practice, the smallest integration time of a multiple-dwell acquisition system is much smaller than the dwell time of a single-dwell acquisition system. This permits dismissal of an incorrect alignment much earlier than would be possible in a single-dwell system. The ability to reject false code alignments much earlier, produces a considerable reduction in acquisition time. Figure 3.3 illustrates the *M*-dwell serial-search acquisition system. The *M* stands for the (maximum) number of times a threshold test is performed during each cell search.

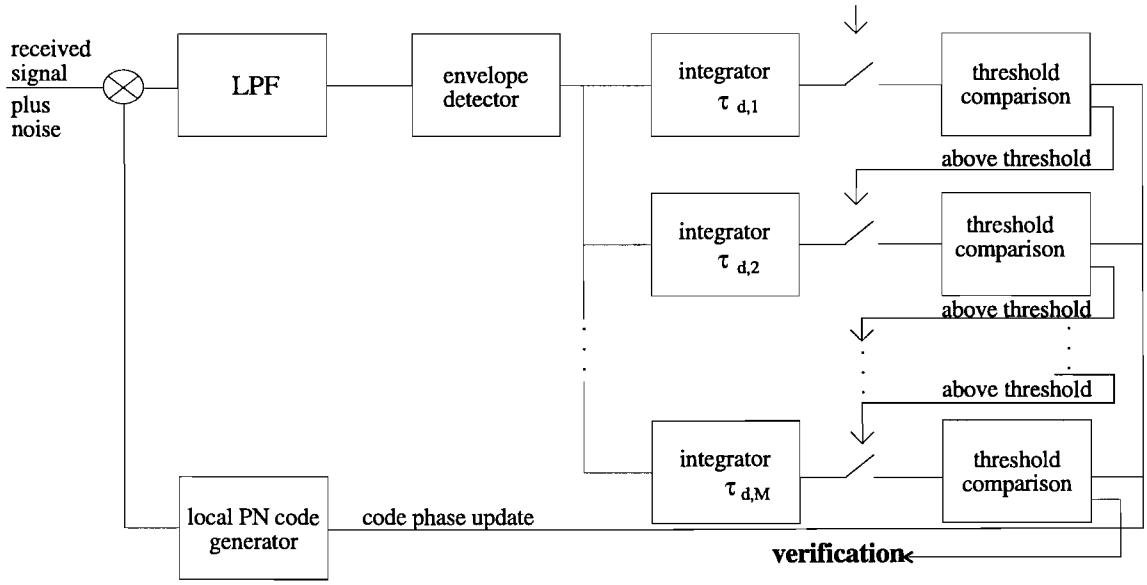


Figure 3.3: Block diagram of the M-dwell serial-search acquisition system.

The system can be characterized by M dwell times $\tau_{d,1}$ to $\tau_{d,M}$, M false alarm probabilities $P_{FA,1}$ to $P_{FA,M}$, and M detection probabilities $P_{D,1}$ to $P_{D,M}$. On the assumption that the detector dwell times are ordered such that

$$\tau_{d,1} \leq \tau_{d,2} \leq \tau_{d,3} \leq \cdots \leq \tau_{d,M} , \tag{3.6}$$

the decision to continue or stop the search at the present cell is made by sequentially examining the M detector outputs and applying to the following algorithm:

1) if all of the M detectors, tested in succession, indicate that the present cell is correct, i.e., each produces a threshold crossing, then the decision is made to stop the search;
2) if any one detector fails to indicate that the present cell is correct, i.e., it fails to produce a threshold crossing, then the decision is made to continue the search and the next cell will be examined. Thus, as soon as one detector indicates that the codes are misaligned, the search may move on without waiting for the decision of the other detectors.

The minimum time to search a given cell is $\tau_{d,1}$, whereas the maximum time is $\tau_{d,M}$. Herein lies the power of the M-dwell acquisition system, namely, that most of the cells can be dismissed after a dwell time $\tau_{d,k}$, with $k \ll M$, whereas the single-dwell acquisition system requires that each cell be examined for a time equivalent to $\tau_{d,M}$. The choice between the single-dwell and the multiple-dwell system must be a compromise between acquisition time and complexity of implementation.

3.2.3.1 Multiple-dwell acquisition-time performance

The difference in acquisition-time performance between the multiple-dwell and the single-dwell serial-search system is that, for the multiple-dwell system, the time required to step to the next cell is varying from cell to cell. The analysis for calculating the mean and variance of the multiple-dwell serial-search acquisition system is analogous to the calculation of the acquisition-time performance of the single-dwell system and is described in [5].

The conditional detection probability $P_{D,i}$ corresponds to the probability that, for the cell containing signal plus noise, the i-th integrate-and-dump output Z_i exceeds its threshold conditioned on $Z_1, Z_2, .., Z_{i-1}$ all having exceeded their respective thresholds. In mathematical terms, letting η_i denote the i-th threshold level, then

$$P_{D,i} = Pr\{ Z_i > \eta_i \mid Z_1 > \eta_1, Z_2 > \eta_2, .., Z_{i-1} > \eta_{i-1} \}. \tag{3.7}$$

Similarly, for a cell containing only noise, the conditional false alarm probability $P_{FA,i}$ is defined identically to Equation (3.7)

$$P_{FA,i} = Pr\{ Z_i > \eta_i \mid Z_1 > \eta_1, Z_2 > \eta_2, .., Z_{i-1} > \eta_{i-1} \}. \tag{3.8}$$

The penalty time T_p of the multiple-dwell serial-search method is defined by $T_p=K_M\tau_{d,M}$.

Similar to the calculation in Section 3.2.2.1, the mean and variance of the acquisition time can be calculated. This results in [5]

$$T_{ACQ,\,mean} = \frac{(2-P_D)q\sum_{j=1}^{M}\left[\left(\prod_{i=1}^{j}P_{FA,i}\right)(\tau_{d,j}-\tau_{d,j-1}) + K_M P_{FA}\delta_{jM}(\tau_{d,M}-\tau_{d,M-1})\right]}{2P_D} \tag{3.9}$$

where $P_D = \prod_{i=1}^{M} P_{D,i}$, $P_{FA} = \prod_{i=1}^{M} P_{FA,i}$, and δ_{jM} is the Kronecker delta function, which is 1 for $j=M$ and zero elsewhere. The variance of the acquisition time is

$$\sigma^2_{T_{ACQ}} = q^2\left\{\sum_{j=1}^{M}\left[\left(\prod_{i=1}^{j-1}P_{FA,i}\right)(\tau_{d,j}-\tau_{d,j-1})+K_M P_{FA}\delta_{jM}(\tau_{d,M}-\tau_{d,M-1})\right]\right\}^2\cdot\left(\frac{1}{12}+\frac{1}{P_D^{\,2}}-\frac{1}{P_D}\right) \tag{3.10}$$

Of course, the multiple-dwell serial-search acquisition system is much more complex than the single-dwell system. The dwell times must be chosen correctly, and according to these dwell times, the thresholds of the detector(s) must be set optimally depending on many system parameters. However, the system shown in Figure 3.3 represents only a conceptual system. In practice, the M integrate-and-dumps are realized by a single continuous-time integrator whose output is sequentially sampled (but not dumped) at time instants $t=\tau_{d,1},\tau_{d,2},..,\tau_{d,i}$ depending, as above, on the outcomes of the first i-1 threshold comparisons. This single integrator is reset only after the decision is made to search the next cell.

It is impossible to give an example of the multiple-dwell acquisition-time performance compared to the single-dwell acquisition-time performance. This is because the *conditional* detection probabilities and the *conditional* false-alarm probabilities can not be computed, given the *unconditional* probabilities. However, it is shown in [5] that the acquisition-time performance of the multiple-dwell system is always better than the performance of a single-dwell acquisition system. Even the simplest implementation of a multiple-dwell acquisition system, i.e., a double-dwell acquisition system, yields an acquisition-time performance that is better under the same conditions.

3.3 Acquisition using matched filtering

If *c(t)* is any physical waveform, then a filter which is matched to *c(t)* is, by definition, one with impulse response

$$h(t) = k\,c(\tau - t), \tag{3.11}$$

where k and τ are arbitrary constants. This means that a matched filter is a device of which the impulse response is the time-reversed and delayed replica of the input signal, that the filter is matched to. If *c(t)* is a pseudo-noise code, the filter impulse response is the time-reversed and delayed pseudo-noise code. The maximum output will occur for a specific phase of the input signal. This maximum can be sensed and used to start the local code generator at the appropriate phase. In this way, synchronization is accomplished.

The pseudo-noise matched filter can be implemented in many ways. For BPSK modulated pseudo-noise signals, surface acoustic wave (SAW) devices and charge coupled devices (CCD) are commonly used. These are mostly implemented in ASICs. For baseband applications, tapped delay-lines are commonly used. Because the implementation of the acquisition method is on a digital signal processor, the rest of this section considers only digital matched filters.

A general baseband digital matched filter for pseudo-noise signals is illustrated in Figure 3.4.

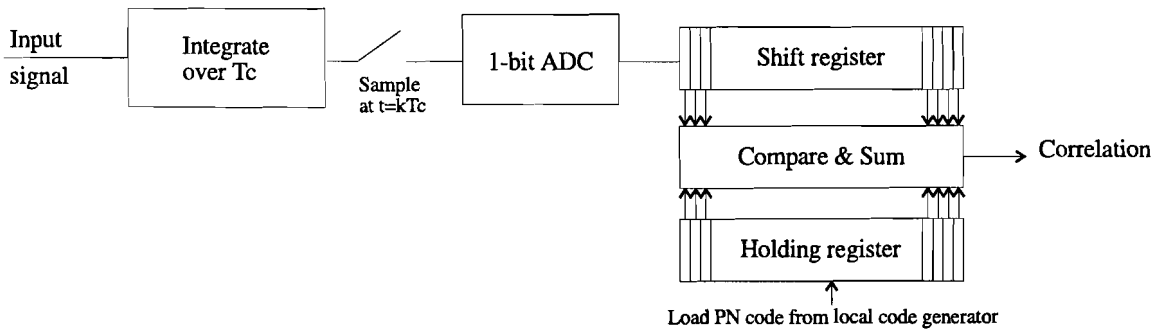


Figure 3.4: A tapped delay-line digital matched filter.

The operation of the digital matched filter is as follows. The PN code is stored in the holding register. At each clock pulse, the input signal is sampled (once per chip, as an example) and loaded into the upper shift register. The contents of the two shift registers are multiplied in pairs and the products are added. The output of the matched filter is the autocorrelation of the pseudo-noise sequence plus noise. The maximum output is sensed by a threshold detector. If the threshold is exceeded, a verification is started. The reference code in the shift register is shifted one position to the right and again a multiplication of

the contents of the two shift registers is performed and the products are added. This is done after each clock pulse during a period $T_p=KT_c$. T_p is called the verification or penalty time. If the threshold is exceeded more than B out of K times during the verification procedure, the system hands over to tracking. If not, the local code reference is held fixed again and the search continues.

3.3.1 Acquisition-time performance using matched filtering

The calculation of mean and variance of the acquisition time for a digital matched filter is given in [5]. It goes beyond the aim of this chapter to describe this analysis, and the results are meaningless without the analysis. Therefore, to give an indication of the acquisition-time performance of the digital matched filter, a comparison is made with the single-dwell serial-search acquisition method. For this analysis, it is assumed that the dwell time is the full code period, $\tau_d=NT_c$, and the threshold setting of the detector is identical. Also, the acquisition time is considered without the presence of any frequency uncertainty.

Because the thresholds are identical, the detection probability and the false alarm probability both are the same for the two techniques, and the only difference is the time to acquire the correct code phase. To compare the acquisition-time performance of a single-dwell acquisition system and a digital matched filter acquisition system, Figure 3.1 is referenced. The matched filter can detect in a single chip-time T_c whether the correct code phase is detected or not, whereas the single-dwell acquisition system uses the full code-length period NT_c to detect if the code phase is correct. Thus, the mean acquisition time of the digital matched filter receiver will be N times smaller than the acquisition time for the single-dwell serial-search technique. The variance will be N^2 times smaller.

When the dwell time τ_d is not the whole code period but a fraction of it, the detection probability will be smaller and the false alarm probability will be higher. However, if we assume that these probabilities will stay the same, the mean acquisition time of the matched filter receiver is τ_d/T_c times smaller than for single-dwell serial search. Otherwise, the ratio of the acquisition time of the single-dwell acquisition system and the acquisition time of the system using matched filtering will be higher.

In the presence of frequency uncertainty, the mean and variance of the acquisition time can be corrected for this influence in the same way they are corrected for the single-dwell serial search. The correction is given in Equations (3.4) and (3.5).

4 System implementation

4.1 Introduction

This chapter discusses the picoterminal system and the baseband transmitter and receiver, which are part of it. Furthermore, the implementation of a baseband acquisition system is discussed. The hardware implementation of this system is further explained in Chapter 5. The software is explained in Chapter 7.

4.2 Picoterminal implementation

As discussed in the general introduction, the multiple-access technique most suitable for a picoterminal satellite communication system is spread-spectrum multiple access. The most commonly used modulation technique in spread-spectrum satellite communications is BPSK direct-sequence spread spectrum, as discussed in Section 2.4.2. It is the simplest form of spread-spectrum modulation and has good performance. For this reason it is also applied in the picoterminal system.

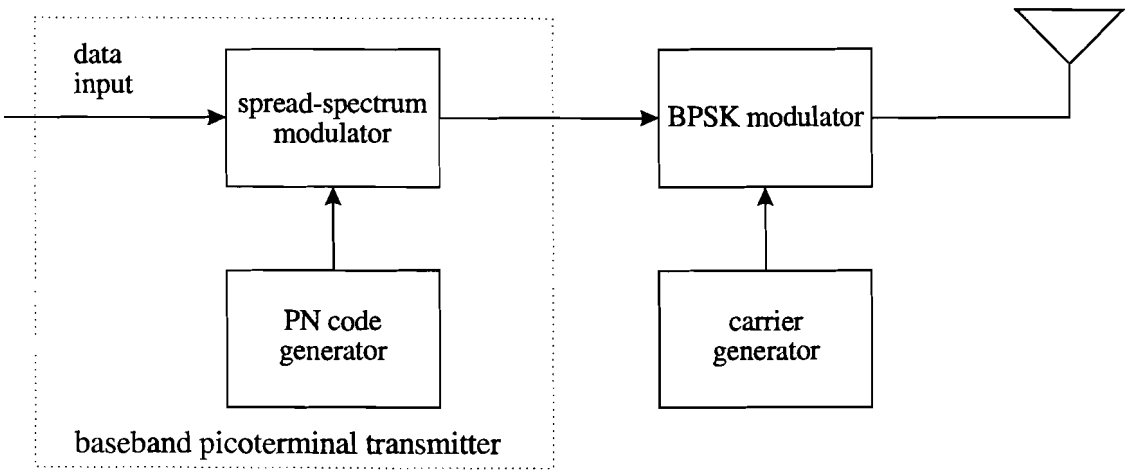


Figure 4.1: The picoterminal transmitter.

Figure 4.1 illustrates the schematic diagram of the picoterminal transmitter. The transmitter consists of a spread-spectrum modulator, which multiplies the data signal with the spreading code, and a BPSK modulator which modulates the spreaded data signal. As discussed in Section 2.4.2, multiplication of the data signal with the spreading code and BPSK modulation can be performed in random order. The picoterminal transmitter first

performs multiplication of the data signal with the spreading code, followed by BPSK modulation of the spreaded data signal. This implies that spreading of the data signal is performed by the baseband picoterminal transmitter, as illustrated in Figure 4.1. The advantage of this implementation is discussed later in this section.

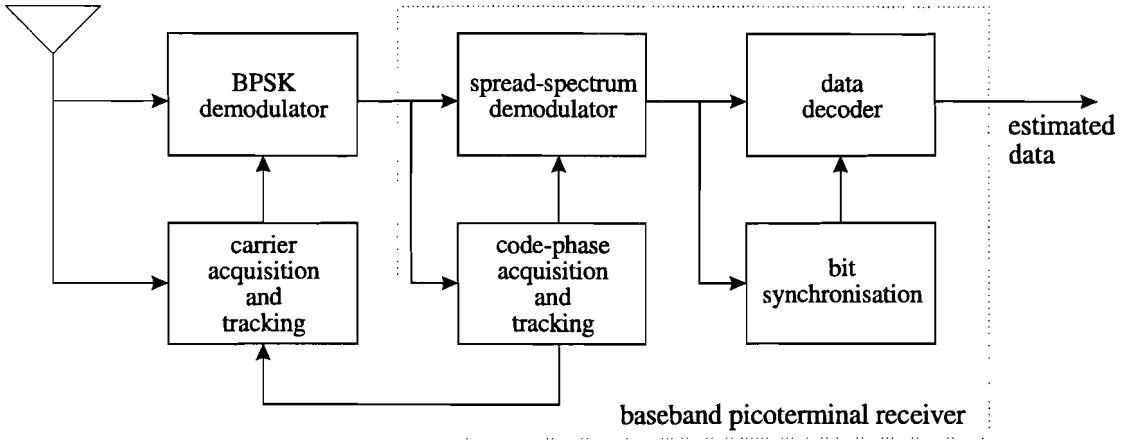


Figure 4.2: The picoterminal receiver.

The schematic diagram of the picoterminal receiver is shown in Figure 4.2. The picoterminal receiver consists of a BPSK demodulator, that is synchronized to the received waveform by the carrier-acquisition and tracking circuit. The BPSK-demodulated signal is despreaded by the spread-spectrum demodulator through multiplication with the locally generated spreading code. The local spreading code is synchronized to the received signal by the code-phase acquisition and tracking circuit. Because the correct code phase can only be acquired if both the carrier frequency as well as the local spreading code are synchronized to the received signal, the code-phase acquisition and tracking circuit provides feedback to the carrier acquisition and tracking circuit. When synchronization is accomplished, the demodulated signal can be decoded. Decoding is performed using a bit-synchronization circuit. As discussed in Section 2.4.2, BPSK demodulation and multiplication of the received signal can also be performed in reversed order. In the picoterminal receiver, BPSK demodulation of the received signal is performed before the signal is despreaded. This enables the implementation of the spread-spectrum demodulator and the code-phase acquisition and tracking circuit in a baseband system. Implementation of the spread-spectrum modulator and demodulator in a baseband system has a considerable advantage. This advantage is discussed in the following.

As discussed in Section 2.7, the main problems that have to be coped with in a direct-sequence spread-spectrum system are acquisition and tracking of the local carrier and the locally generated spreading code to the received signal. The baseband picoterminal receiver has to acquire and to track the correct code phase, and to despread the received signal. To acquire and to track the correct code phase, many correlations have to be

performed. The faster these correlations are performed, the better the acquisition-time performance of the system. Implementation of the spread-spectrum modulator and demodulator in the baseband system provides the implementation of the baseband picoterminal transmitter and receiver on an electronic circuit based on a digital signal processor. These processors have an architecture that is optimized to calculate multiplications and correlations. This means that correlations can be computed very fast, which yields a good acquisition-time performance.

To come to the implementation of a baseband picoterminal system, the hardware of a baseband picoterminal transceiver was designed. The hardware is based on the Texas Instruments TMS320C25 digital signal processor, which operates at the maximum clock frequency of 50 MHz. The electronic circuit operates at the maximum clock frequency to yield its maximum acquisition and tracking performance. This specific processor is used since it is a frequently used digital signal processor and at the Telecommunications Division there was already some experience in applying it.

4.3 Implementation of the acquisition system

The hardware was designed to implement the complete baseband picoterminal transmitter and receiver. However, within the time constraints, only the acquisition system was implemented.

In Section 3.2, two acquisition methods were presented. These are serial search acquisition and acquisition using matched filtering. A matched filter is mostly implemented in hardware, because implementation in software requires processing speeds that can not be realized with available digital signal processors. Therefore, the serial-search acquisition method was implemented.

The serial-search method discussed in Section 3.2 is characterized by stepping from cell to cell in the phase/frequency uncertainty region illustrated in Figure 3.1. Two methods to detect the correct cell were presented. These are the single-dwell and the multiple-dwell serial-search method. The single-dwell method is the simplest one. In fact, the multiple-dwell method is an extension of the single-dwell method. Only more dwell times and detection thresholds are being used, which makes the method more complex. Therefore, the single-dwell serial-search method is implemented in first instance. Knowledge of the performance of this method can be used for the implementation of a multiple-dwell system.

Parameters and conditions which influence the performance of the acquisition system are for example the detection threshold and the signal-to-noise or signal-to-interference ratio. Although for a baseband system no theory of the influence of these parameters and conditions on the acquisition-time performance is available (as discussed in Section 3.2.2.1), these influences can be measured to gain knowledge of the behaviour of the single-dwell acquisition system though.

The single-dwell serial-search acquisition method was implemented in the baseband transceiver software. The acquisition system consists of two transceivers, one configured as a transmitter and one as a receiver. With this system, acquisition-time measurements can be performed for various signal-to-interference ratios. For this purpose, the transmitter not only transmits a spreaded data signal, it generates a spreaded data signal embedded in the spreaded signals of other channel users, each having its own unique code. The spreading codes used are maximal-length sequences. Properties and generation of these sequences are discussed in Section 2.3. The number of other channel users can be varied. In this way, the acquisition system can be tested under various interference conditions, and the acquisition-time performance can be measured as a function of the signal-to-interference ratio and as a function of the detection threshold. In first instance, the acquisition-time performance is measured in the absence of frequency uncertainty. To assure this, a synchronous system is implemented.

To provide the baseband transmitter and receiver to communicate with a personal computer, a communications interface was implemented on the transceiver. The acquisition-time measurements are controlled by the personal computer. It sends the appropriate data for a measurement to the transmitter and receiver and receives the measurement data from the receiver when the correct code phase has been acquired. The measurement procedure is fully automatized. Many acquisition-time measurements can be performed automatically, and thus the acquisition behaviour of the system can be tested.

4.4 Conclusions

The single-dwell serial-search acquisition method was implemented in a baseband system. With this system, acquisition-time measurements can be performed automatically. This provides us to test the acquisition-time performance of the single-dwell serial-search method as a function of the detection threshold and the signal-to-interference ratio.

The hardware was designed to implement a complete baseband picoterminal. Therefore, one can not only test the acquisition system, in future also the tracking behaviour and the bit-error rate can be studied.

The hardware of the baseband picoterminal is described in Chapter 5. More details on the design program, components used, and the printed circuit board design can be found in Appendices 1 and 2. The implementation of the acquisition method in software is discussed in Chapter 7. Chapter 6 discusses the communications interface implemented on the transceiver. In Chapter 8, the measurement procedure is explained and results are presented. Finally, Chapter 9 discusses the conclusions and recommendations.

5 Transceiver hardware

5.1 Introduction

This chapter describes the transceiver hardware developed. Section 5.2 gives a schematic diagram of the designed electronic circuit, which is further explained in Sections 5.3 to 5.10. Sections 5.11 and 5.12 explain the transmitter and receiver configurations, respectively. Section 5.13 describes the print lay-out and EMC considerations, and Section 5.14 describes the hardware errors made. Finally, Section 5.15 finishes this chapter with the conclusions and recommendations.

5.2 Schematic diagram of the transceiver hardware

The electronic circuit of the baseband transceiver can be divided into several functional blocks. Figure 5.1 illustrates the schematic diagram of the designed electronic circuit. The functional blocks are listed below.

- digital signal processor
- oscillator (OSC.)
- reset circuit (RESET)
- wait-state generator (WSG)
- memory
- I/O selection
- digital-to-analog converter (DAC), output buffer, and transmit clock
- analog-to-digital converter (ADC)
- communications interface (COMM. INT.) and DIP-switches

The communications interface and DIP-switches are described in Chapter 6.

In the first instance, the intention was to build separate transmitter and receiver circuits. In fact, the transceiver circuit designed is a direct-sequence spread-spectrum transmitter with a small extension. If the transmitter and receiver were build separately, a large part of the system would be the same. Therefore, the transmitter circuit extended with an analog-to-digital converter (ADC) forms the transceiver circuit. The circuit can be configured as a

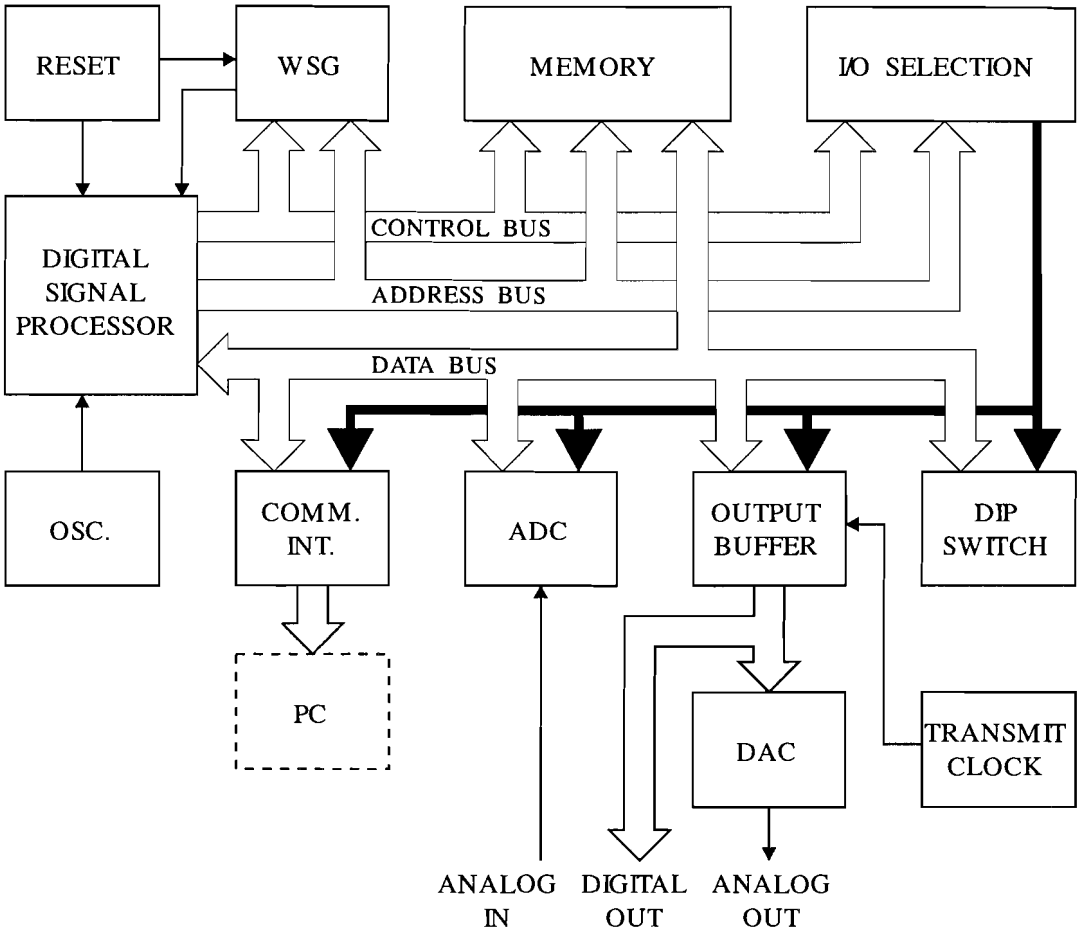


Figure 5.1: Schematic diagram of the transceiver hardware.

transmitter or a receiver by setting jumpers. To extend the receiver circuit, a printed circuit board can be plugged into the IC-socket of the ADC. Further details on the transmitter and receiver configuration are explained in Sections 5.11 and 5.12, respectively.

The transceiver is designed as a prototype to test as many characteristics as possible. With jumper settings, the transceiver configuration can be changed in several ways. Most of the relevant control, input, and output signals are available at connector pins to create an optimal test device.

Appendix 1 comprises the complete electronic circuit designed. Also jumper settings and connector pins are listed there. Data sheets of several components can be found in Appendix 2. Signals and pins denoted with a '*' are active low. Numbers ending with 'b' or 'h' are binary or hexadecimal, respectively.

5.3 The digital signal processor

The designed electronic circuit is based on the Texas Instruments TMS320C25-50 digital signal processor. Relevant key features of this processor are listed in Table 5.1 [6].

Table 5.1: Key features of the TMS320C25-50.

- clock frequency up to 50 MHz (instruction cycle 80 ns)
- 16-bit address-bus and data-bus
- 32-bit ALU/accumulator
- 544 words programmable on-chip data/program memory
- 128k words total data/program memory space
- 16 input and 16 output ports
- 3 prioritised hardware interrupts
- wait-states for communication to slower off-chip memories/peripherals
- 16x16-bit parallel multiplier with 32-bit product
- single cycle multiply/accumulate instructions

Specifications of this processor and requirements for memories and peripherals will be considered in the following sections. Program memory, data memory, and the I/O ports are referred to as the 3 external spaces, which have their own select signals.

5.4 Oscillator circuit

Figure 5.2 shows the digital signal processor with oscillator circuit. The TMS320C25-50 can use either its internal oscillator or an external frequency source for a clock. The internal oscillator is enabled by connecting a crystal across the X1 and X2/CLKIN pins. Crystals of 20 MHz and below can be used for the internal oscillator. They are oscillating in the fundamental mode. Operation above 20 MHz requires an external oscillator, for example a circuit with a crystal oscillating at its third overtone, i.e., a third-overtone oscillator. An external oscillator can be used by connecting the oscillator output directly to the X2/CLKIN-pin, with the X1-pin left unconnected.

Operation at 50 MHz requires a third-overtone oscillator [6, pg.6-5] or a crystal oscillator. A crystal oscillator is a single device in a DIP-package just requiring a supply voltage for operation. The output is a TTL-compatible square-wave with good frequency stability. In the circuit designed a 50 MHz crystal oscillator (U13) is used, because it is cheap, it saves some space on the printed circuit board, and it does not need any adjustments. The output signal of the crystal oscillator is stable 10 ms (maximum) after power-up. For test

purposes requiring a lower clock frequency, the internal oscillator can be enabled. Selection between the two oscillators is made with jumper JP8. By connecting pins 2 and 3, the crystal oscillator is selected. The internal oscillator is selected by connecting pins 1 and 2 of jumper JP8. Appendix 2 comprises data sheets of the crystal oscillator.

5.5 Reset circuit

For proper system initialization, a reset signal must be applied for at least three machine cycles. A machine cycle is defined as four clock cycles. Thus, for a TMS320C25-50 operating at 50 MHz, the reset pulse must be low for at least 240 ns. Upon power-up, it can take 10 ms before the crystal oscillator reaches a stable operating state. Therefore, at power-up, the reset circuit should generate a low pulse on the reset-line until the oscillator is stable. Figure 5.2 shows the reset circuit together with the digital signal processor and the oscillator circuit. The reset circuit used can also generate a reset pulse by pushing a reset button (S1). The duration of the reset pulse is approximately 17 ms, depending on R5, C1, and the low-to-high threshold voltage of the Schmitt-trigger inverter (U31).

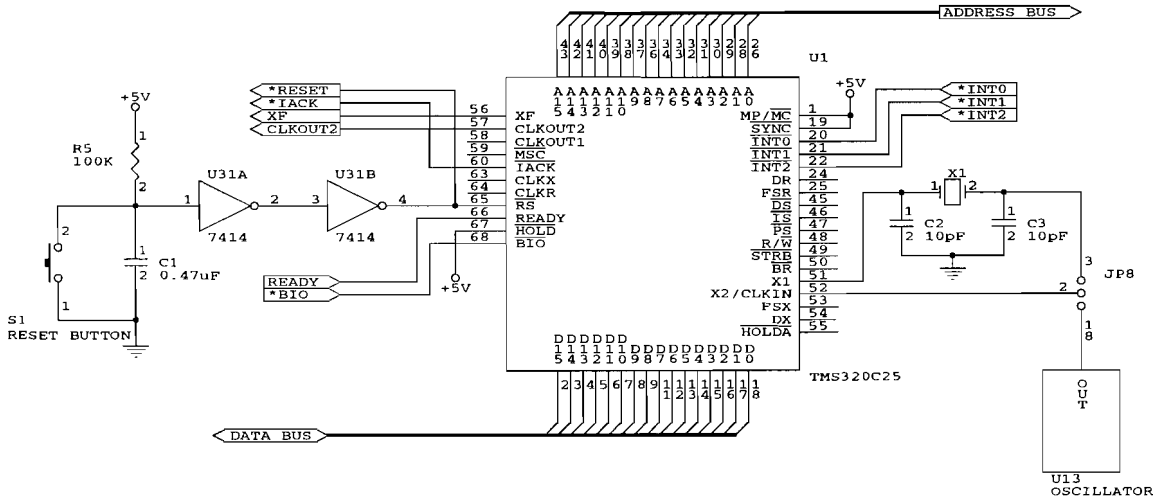


Figure 5.2: The TMS320C25-50 with oscillator and reset circuit.

5.6 The wait-state generator

One of the features of the TMS320C25-50 is the capability to use wait-states for communication to slower off-chip memories and peripherals. Their access can be extended with one or more machine cycles. The number of cycles in a memory or I/O access is determined by the state of the READY-input of the processor at a certain time. A wait-

state generator generates this READY-signal as a function of several address lines and program, data, and I/O select signals. The program, data, and I/O select signals (*PS, *DS, and *IS, respectively), together with the R/*W and *STRB signals are referred to as interface control signals or control bus.

The timing requirement for generation of the READY-signal is specified according to Figure 5.3.

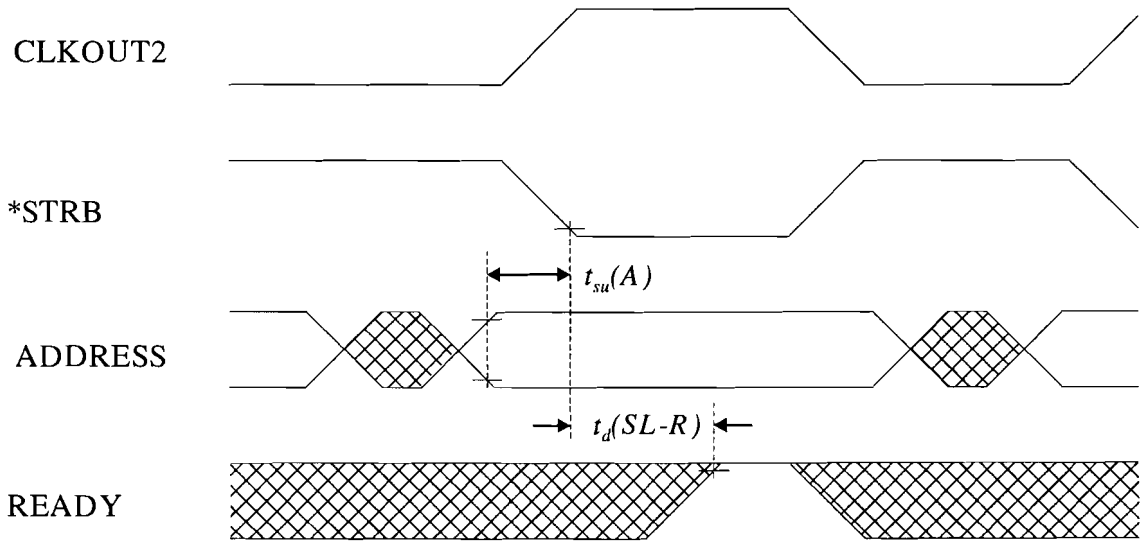


Figure 5.3: READY-timing requirement.

The READY-signal must be valid no later than $t_{su}(A) + t_d(SL\text{-}R)$ after the address-bus and interface control signals (except *STRB) become valid. This evaluates to a maximum of

$$t_{su}(A) + t_d(SL-R) = 9 \text{ ns} \tag{5.1}$$

for a TMS320C25-50 operating with an input clock frequency of 50 MHz [8,pg.56].

Figure 5.4 shows the wait-state generator designed. It is designed according to the circuit shown in [6,pg.6-18].

To satisfy the timing requirement for the READY-signal denoted in Equation (5.1), the maximum signal delay time in U12B and U11B, and the maximum delay time in U21A and U11B may not exceed 9 ns. For commercial types of TTL NAND-ports, this can only be realized by using Advanced Schottky (AS) TTL logic. The 74AS10 (U12 and U21) and the 74AS20 (U11) each have maximum delay times of 4.5 ns.

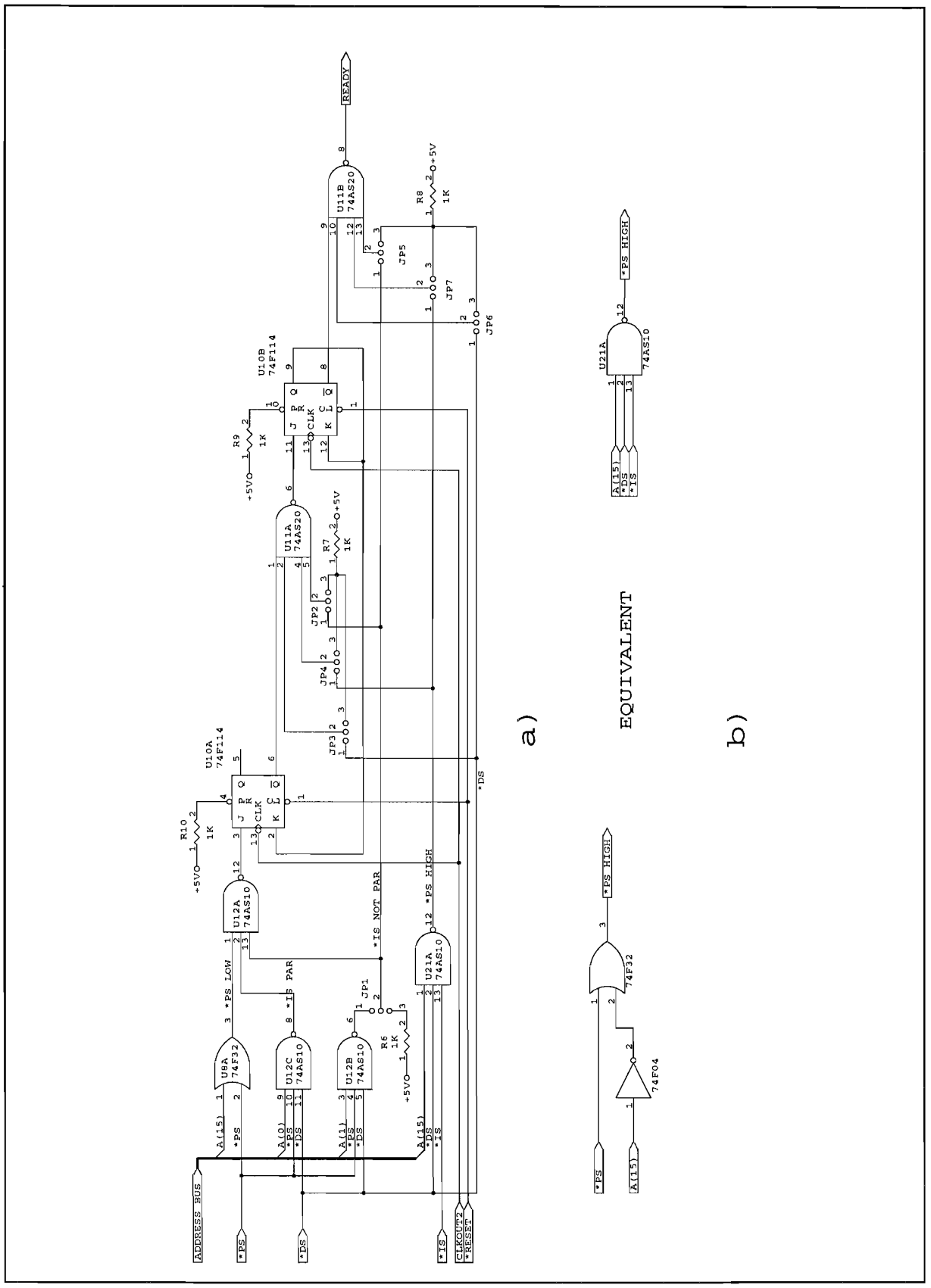


Figure 5.4: a) the wait-state generator designed;
b) *PS, *DS, and *IS are mutually exclusive.

The attentive reader should have noticed that generation of the READY-signal in case of program RAM access is accomplished using address-bus line A(15), *IS, and *DS, and without the use of *PS. Here, the fact that both *IS and *DS are high when making a program memory access is used - i.e., the 3 external spaces are mutually exclusive [8,pg.87]. Using this fact, the two circuits shown in Figure 5.4 b) have the same output. Using the second circuit, one gate delay is saved, which is crucial to satisfy the READY-timing requirement.

The generation of the READY-signal in case of an I/O access is also based on the above principle (parts U12B and U12C). In case of generating the READY-signal for I/O access, another trick is made. To distinguish I/O for parallel port communication and other I/O and to satisfy the READY timing requirement, the address of the parallel port interface is set to 001b and of all other I/O addresses to 010b or 110b. By doing so, the address line A(0) is always opposite to A(1) (only in case of I/O access). Using this knowledge the signals $*IS_{par}$ and $*IS_{not\ par}$ are generated by gates U12B and U12C, respectively.

The number of wait-states can be set by jumpers. Table 5.2 gives an overview of settings of jumpers JP1 to JP7 as a function of the memory or peripheral type and the number of wait-states desired. The number 12 means that pins 1 and 2 of the jumper are connected. The number 23 means that pins 2 and 3 of the jumper are connected.

Table 5.2: Jumper settings for the wait-state generator.

Memory or peripheral	Number of wait-states	Jumper settings JP1 to JP7						
		JP1	JP2	JP3	JP4	JP5	JP6	JP7
EPROM	2	-	-	-	-	-	-	-
Program RAM	0	-	-	-	23	-	-	12
	1	-	-	-	12	-	-	23
Data RAM	0	-	-	23	-	-	12	-
	1	-	-	12	-	-	23	-
I/O Par.prt.	2	-	-	-	-	-	-	-
I/O not Par.prt.	0	23	23	-	-	12	-	-
	1	23	12	-	-	23	-	-
	2	12	23	-	-	23	-	-

5.7 Memory

5.7.1 Introduction

The memory of the transceiver consists of program memory as well as data memory. The TMS320C25-50 provides a total of 544 16-bit words of on-chip data RAM (random access memory), of which 288 words are always data memory, and the remaining 256 words may be configured as either data or program memory.

The external program memory consists of 32k words UV-erasable EPROM (electrically programmable read only memory) and part of the 32k words static RAM. The other part of the 32k words RAM is configured as data memory. The starting configuration uses 16k words RAM program memory and 16k words RAM data memory.

As the access time of the EPROM is 100 ns, it needs 2 wait-states for proper operation. The static RAM is fast enough to run without wait-states. Copying the program from EPROM to RAM directly when initialization is performed, allows full-speed program execution.

Figure 5.5 shows the program and data memory maps [6,pg.3-15]. Figure 5.6 illustrates the TMS320C25-50 with connected EPROMs and static RAMs.

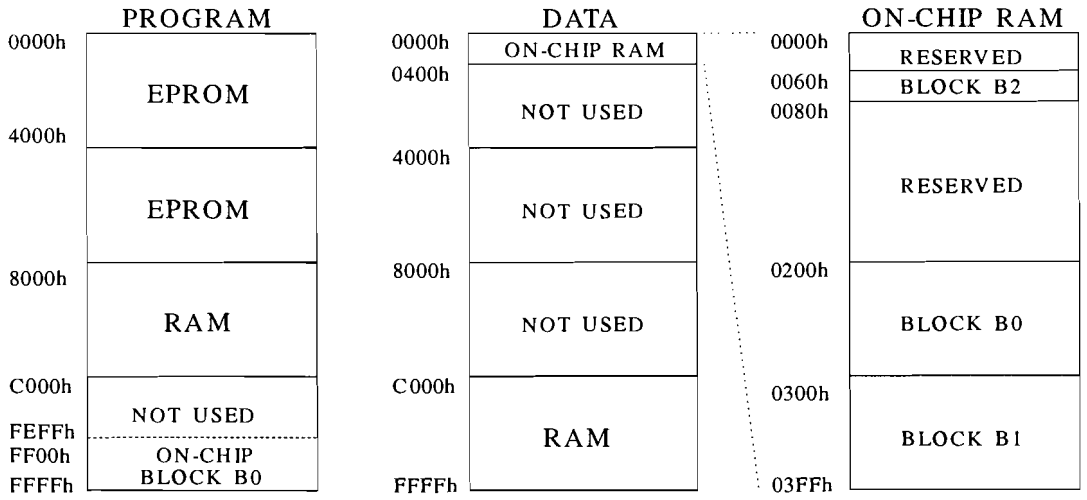


Figure 5.5: Memory maps.

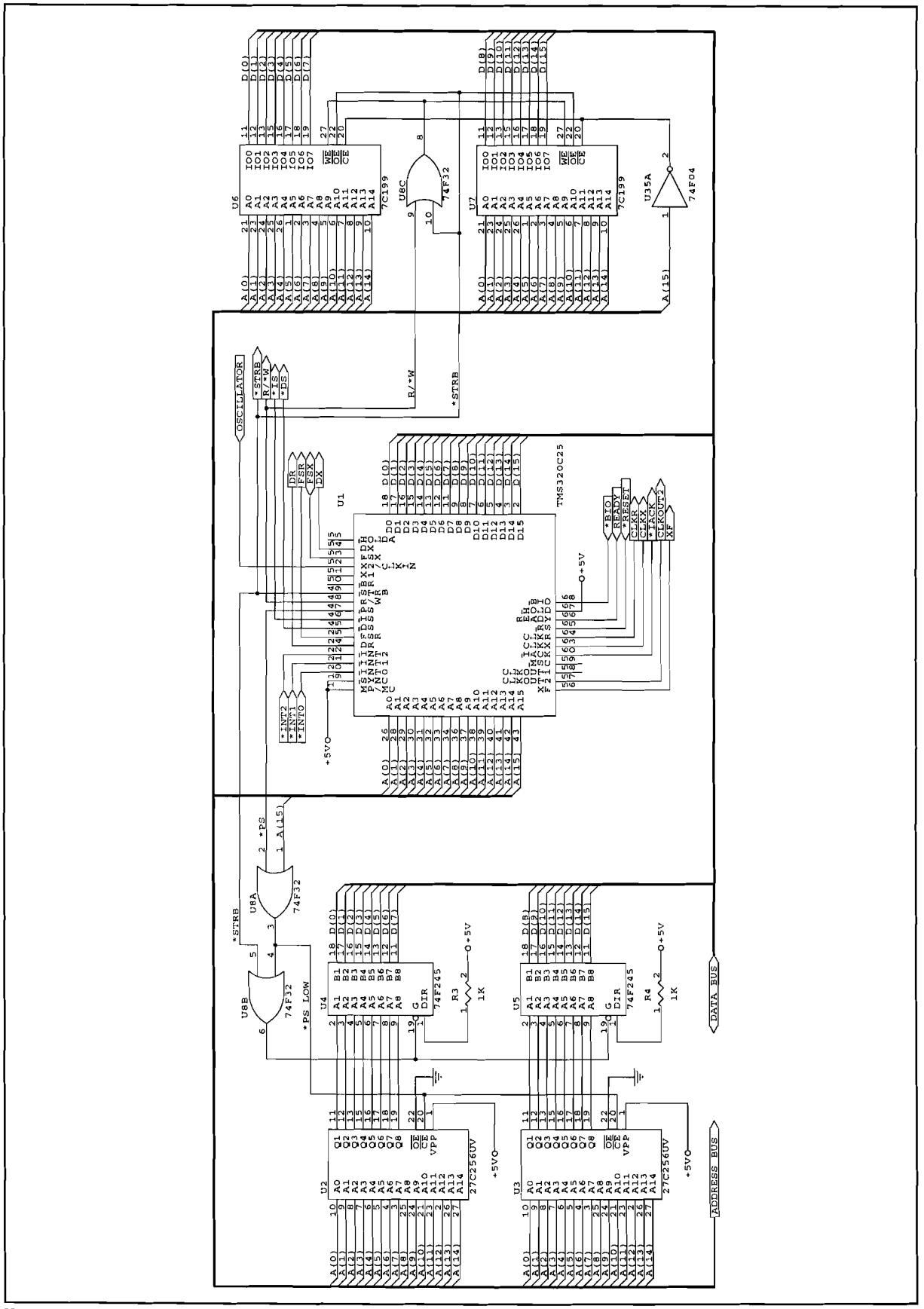


Figure 5.6: The TMS320C25-50 with EPROMs and RAMs.

5.7.2 Program memory

External program memory is divided into two parts. From address 0000h to 7FFFh the 32k words EPROM (U2 and U3) are addressed. U2 drives the low byte of the data bus (D(0) to D(7)) and U3 drives the high byte (D(8) to D(15)).

The output disable time of the EPROMs is too long to guarantee that no bus conflict will occur if an external write cycle follows an EPROM read cycle. This is solved by buffering the data lines with the 74F245 transceiver ICs (U4 and U5).

U2 and U3 are enabled by $*PS_{low}$, which is low when both *PS and A(15) are low. U4 and U5 are enabled by $*PS_{low}$ and *STRB. Because the EPROMs and transceivers are not enabled by the R/*W control signal, it is possible to write to the EPROMs. This must be avoided in software.

The second part of the program memory is static RAM (U6 and U7) addressed from 8000h to BFFFh (initial configuration). The RAM has a maximum access time of 25 ns and can run at full speed with zero wait-states, with the TMS320C25-50 running at 50 MHz. U6 drives the low byte of the data bus (D(0) to D(7)) and U7 drives the high byte (D(8) to D(15)).

U6 and U7 are enabled by *A(15). The read cycle is controlled by the *STRB signal on pin *OE and by the R/*W signal on pin *WE. The write cycle is controlled by the R/*W and *STRB interface control signals on pin *WE. Timing requirements for memory are described in Section 5.7.4.

Program memory can be expanded by another 256 words internal RAM by configuring block B0 as program memory.

5.7.3 Data memory

Data memory consists of 16k words external RAM (initial configuration) and another 544 (or 288) words RAM on the TMS320C25-50. Because U6 and U7 are not addressed or controlled by the *PS or *DS interface control signals, the external RAM can be configured as needed. At first instance, the 32k words can be divided into 16k words program memory and 16k words data memory, but if, e.g., more program memory is needed, the unused data memory can be configured as program memory, or vice versa. The design allows any arbitrary partition of the 32k words RAM into program and data memory. This partition must be made in software. Figure 5.7 shows an overview of

program and data memory maps in practice, with arbitrary partition of the 32k words RAM.

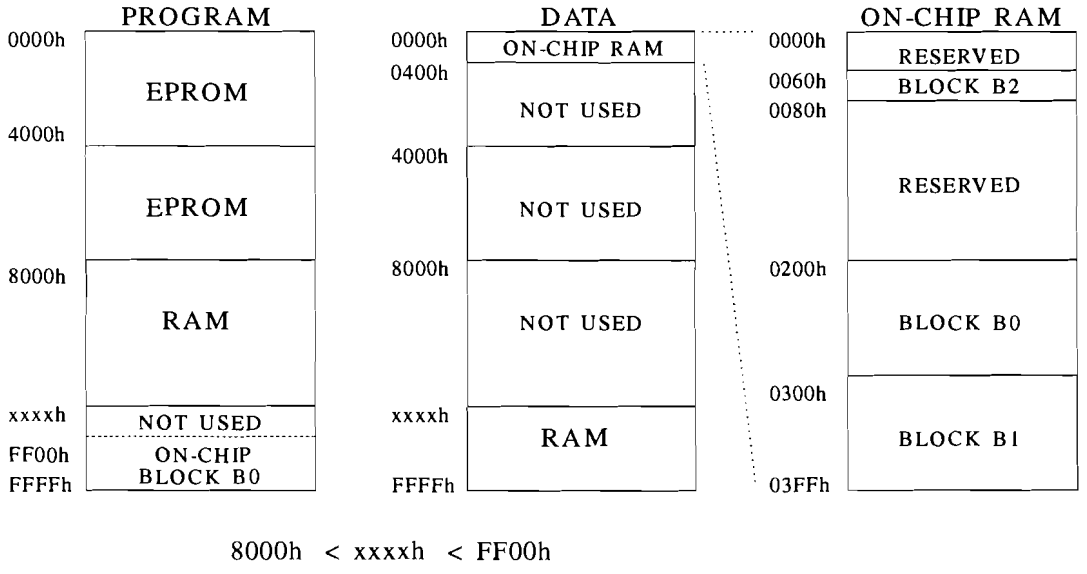


Figure 5.7: Practical memory maps.

The control and addressing signals of the data RAM are the same as for the program RAM, because they are physically the same ICs (U6 and U7).

5.7.4 Memory timing requirements

The TMS320C25-50 can be interfaced to fast static RAM without wait-states. Then, a complete memory access is performed in a single machine cycle. One machine cycle is specified as $4Q$, with Q defined as

$$Q = \frac{1}{f_{osc}}. \tag{5.2}$$

For a TMS320C25-50 operating at 50 MHz, Q evaluates to Q = 20 ns and a machine cycle to $4Q$ = 80 ns. Two key memory device specifications for the read operation of a static RAM are access time from address valid and access time from chip select and/or output enable. These key TMS320C25-50 timing requirements are specified by $t_a(A)$ and $t_a(SL)$, respectively. For a write operation, an important requirement is $t_{su}(D)W$, the time data is valid before *STRB goes high.

In Figure 5.8 and Figure 5.9, respectively, the memory read cycle and the memory write cycle are illustrated and relevant delay and access times are indicated. Using these timing diagrams, timing requirements and component specifications are now being compared [7 and Appendix 2].

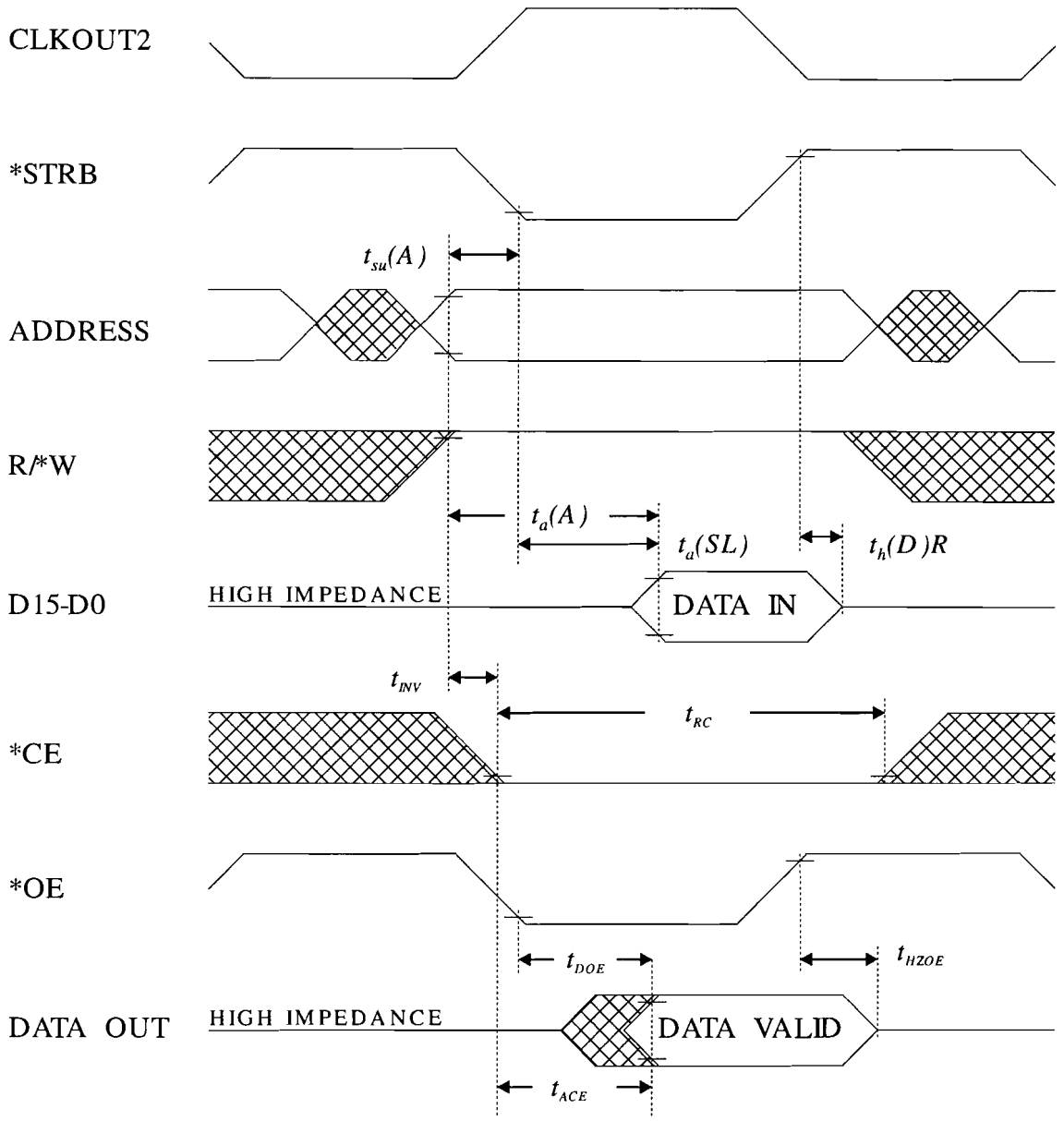


Figure 5.8: Memory-read timing.

- $t_a(A)$ is the access time of the RAM after address becomes valid, and must be less than

$$t_a(A)_{\max} = 3Q - 30 + n \cdot 4Q = 30 \text{ ns} \quad (n=0), \tag{5.3}$$

for a TMS320C25-50 running at a 50 MHz clock frequency [7,pg.36], with n the number of wait-states. According to the specifications of the static RAM and inverter (U35A), data is available no later than

$$t_{INV,\,\max} + t_{ACE,\,\max} = 4.3 + 25 = 29.3 \text{ ns}. \tag{5.4}$$

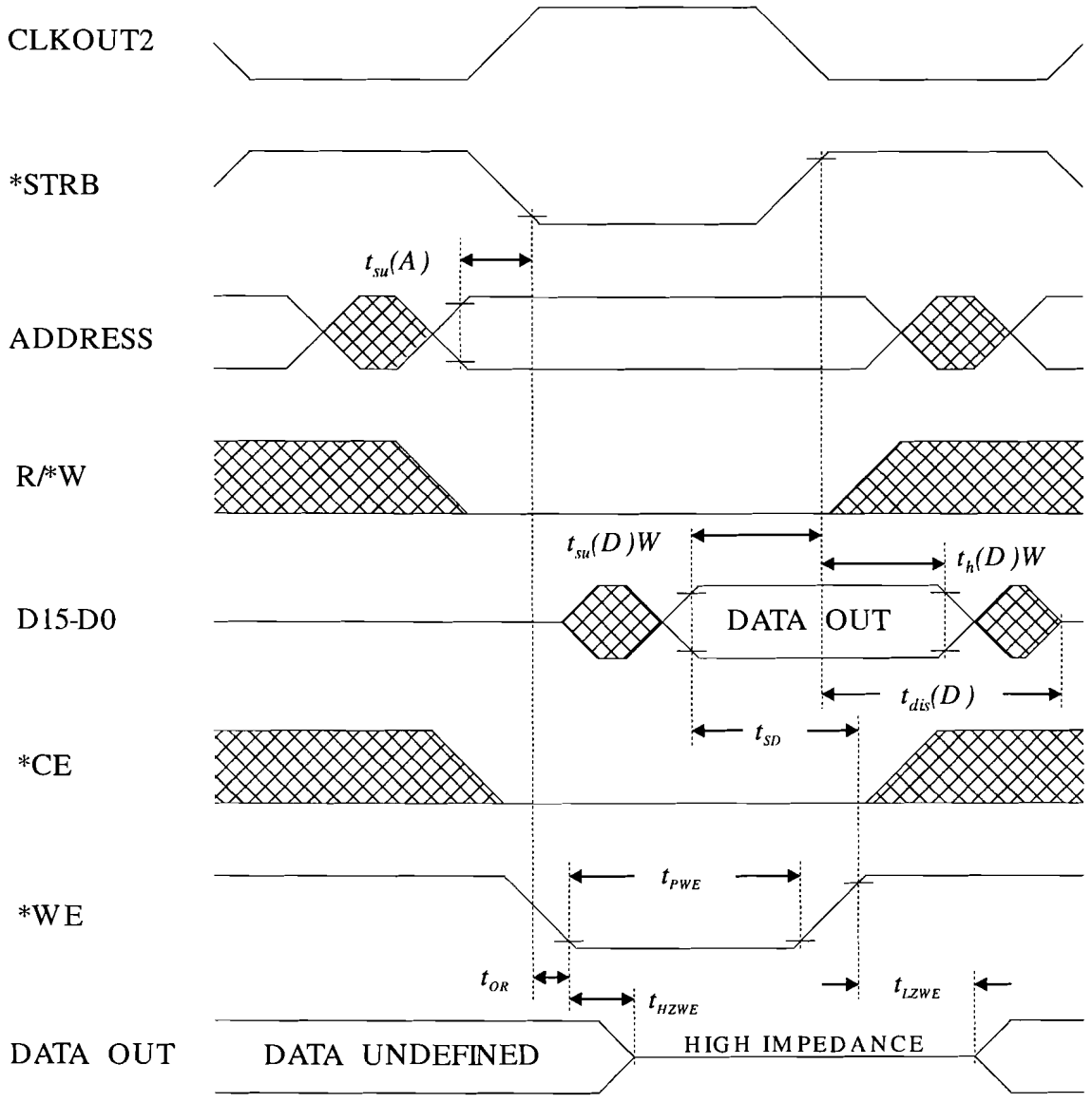


Figure 5.9: Memory-write timing.

- $t_a(SL)$ is the access time of the RAM after *STRB goes low, and must be less than

$$t_a(SL)_{\max} = 2Q - 20 + n \cdot 4Q = 20 \text{ ns} \qquad (n=0), \tag{5.5}$$

for a TMS320C25-50 running at a 50 MHz clock frequency [8,pg.82]. According to the specifications of the static RAM, data is available no later than

$$t_{DOE,\,\max} = 15 \text{ ns}. \tag{5.6}$$

- $t_{su}(D)W$ is the time data on the data bus is valid before *STRB goes high. This is a timing requirement the TMS320C25-50 has to satisfy. $t_{su}(D)W$ must exceed

$$t_{OR,\,\max} + t_{SD,\,\max} = 5.6 + 15 = 20.6 \text{ ns}. \tag{5.7}$$

$t_{su}(D)W_{min}$ is 23 ns, according to the specifications of the digital signal processor.

The specifications of the above considered access times all satisfy the timing requirements of the components used. There are some more timing requirements to satisfy, concerning enable and disable times and cycle times. Concerning the read cycle, the following requirements are satisfied:

- the read cycle time of the static RAM, t_{RC}, must exceed 25 ns. The time *STRB (and thus *OE) is low is at least 35 ns.
- the data read hold time of the external device from *STRB high, $t_h(D)R$, must be at least 0 ns. Because *STRB enables the output of the RAMs, this requirement is always satisfied. The time from *OE high to high impedant output of the RAM, t_{HZOE}, varies from 0 to 13 ns maximum.

Requirements concerning the write cycle are:

- the *WE pulse width, t_{PWE}, must exceed 20 ns. The time *STRB (and thus *WE) is low exceeds 35 ns.
- the data write hold time of the signal processor from *STRB high, $t_h(D)W$, must exceed the delay time of the OR-port. The low-to-high delay time of the OR-port has a maximum of 5.6 ns. $t_h(D)W$ has a minimum of 15 ns.

Furthermore, output disable times must be small enough to avoid bus conflicts if, e.g., a write operation is followed by a read operation. All components are fast enough for these requirements. Thus, all timing requirements are satisfied.

Because the electronic circuit is designed for EPROMs having an access time of 120 ns, two wait-states are required. The maximum access time evaluates to

$$t_a(A)_{\max} = 3Q - 30 + 8Q = 190 \text{ ns}. \tag{5.8}$$

The EPROMs used have a maximum access time of 100 ns. Together with OR-ports U8A and U8B, this evaluates in a maximum access time of 110.6 ns. Thus, normally the EPROMs will operate properly with one wait-state ($t_a(A)_{max} = 110$ ns), but timing with one wait-state is critical. Because the EPROMs are only accessed during initialization when the contents are copied to the static RAMs, only during initialization time could be saved. Thus, two wait-states are acceptable.

5.8 I/O selection

The I/O selection circuit is illustrated in Figure 5.10. I/O selection is accomplished by the 74F138 (U15) 1-of-8 decoder. It accepts three binary weighted inputs A(0), A(1), and R/*W, and when enabled, provides eight active low outputs. The device is enabled by *STRB and by *IS. Of the 16 I/O ports available, only 4 are used. The port addresses are listed in Table 5.3. The reason why these specific addresses are used can be found in Section 5.6.

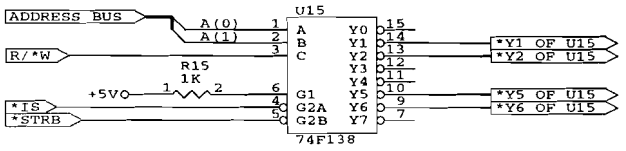


Figure 5.10: I/O-selection circuit.

Table 5.3: I/O port addresses.

Port address	IN operation	OUT operation
1h	Parallel port D(0)-D(7) DIP-switches D(8)-D(15)	Parallel port D(0)-D(7)
2h	D/A converter D(0)-D(11)	A/D converter D(0)-D(13)

5.9 Digital-to-analog converter

As is discussed in Section 5.11, a digital-to-analog converter is needed to provide multi-user simulation. The DAC used is the Analog Devices AD568 (U29). It is a 12-bit monolithic current-output DAC with a settling time to 1 LSB (least significant bit) in 35 ns. Figure 5.11 shows this DAC with an appropriate current-to-voltage converting operational aplifier (U30). The configuration implemented is inverting and has a +5.12 V to -5.12 V voltage output. The settling-time of the buffered output is about 100 ns, which can be optimized by changing capacitor C6. With resistor R27, the offset voltage of the operational amplifier can be adjusted to zero.

As the DAC has no input buffer, an external buffer must be provided. The external buffer is clocked with a transmit-clock signal. The transmit procedure is as follows. On the rising edge of the transmit clock (which is either internal or external), data is clocked into the

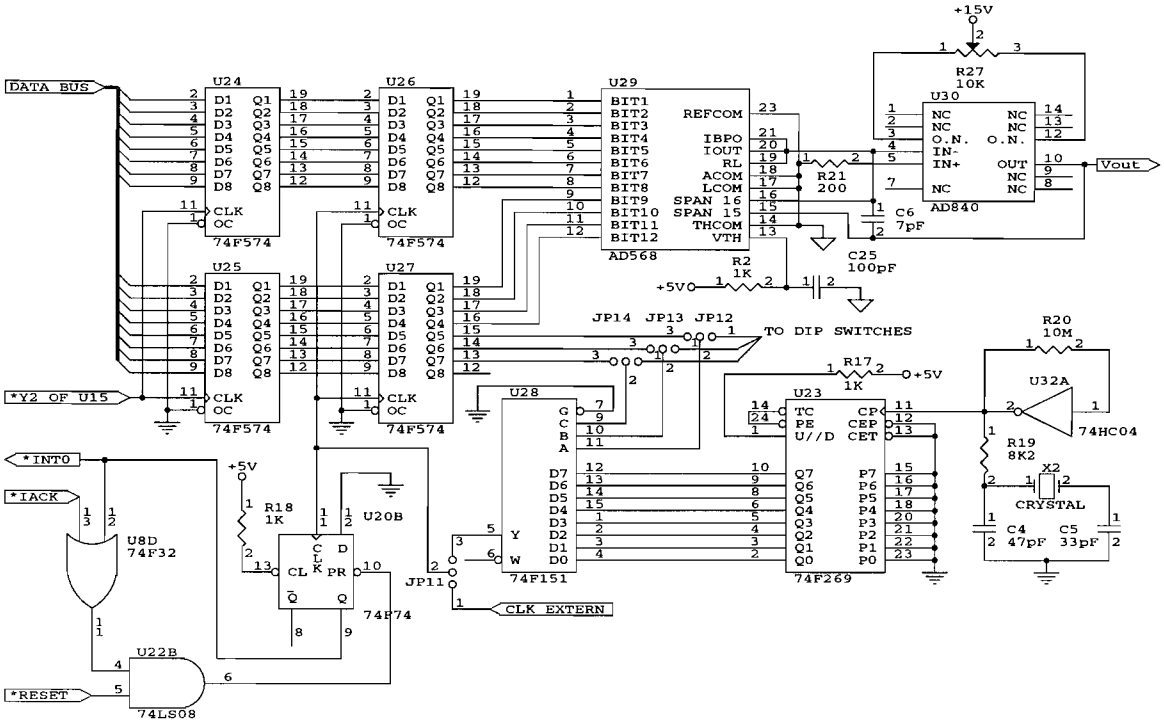


Figure 5.11: Buffered bipolar-voltage output digital-to-analog converter with interrupt-generation circuit and transmit clock.

buffer (U26 and U27) and the buffer output is converted to an analog bipolar voltage within approximately 100 ns. At the same time, an interrupt (*INT0) is generated at the processor. The processor writes the next word to be converted to a second buffer (U24 and U25), which is the input to the first buffer. Then, on the next rising edge of the transmit clock, the whole procedure repeats itself.

The internal transmit clock has an adjustable frequency of 8 kHz to 1.024 MHz, and consists of an oscillator circuit, a binary counter and an 8-bit multiplexer. The oscillator is based on crystal (X2), oscillating at 2.048 MHz, and a CMOS inverter (U32A). This square wave is the input of a 8-bit binary counter (U23), which serves as a divider circuit. The 8 outputs have frequencies from 8 kHz (Q7) to 1.024 MHz (Q0). The 8 input multiplexer (U28) selects one of the outputs of the counter to serve as transmit clock signal. The multiplexer is controlled by three select inputs A, B, and C, which can be set by DIP-switches DIP1 to DIP3 or programmed in software. Three unused outputs of buffer U27 (Q5 to Q7) provide the select inputs which can be programmed. Selection between DIP-switches and software can be made by setting jumpers JP12 to JP14. By connecting pins 1 and 2 of these jumpers, selection is made by DIP-switches; selection by software is made by connecting pins 2 and 3.

The output of the transmitter can also be clocked out by an external clock signal. This can be used if, e.g., a BPSK-modulator is connected to the baseband transmitter. The choice between the internal or external transmit clock is made by jumper JP11 (connection if pins 2 and 3 for internal transmit clock, pins 1 and 2 for external transmit clock). The external clock also generates an interrupt at the processor.

The interrupt, caused by the transmit clock, is generated by clocking a zero into a D-flip flop (U20B). Interrupt 0 (*INT0) is used, because this external interrupt has the highest priority within the signal processor. The interrupt is disabled when the *IACK (interrupt acknowledge signal) output of the processor goes low or if the processor is reset.

5.10 Analog-to-digital converter

For analog-to-digital conversion of the receiver input the Analog Devices AD779 is used. It is a 14-bit analog-to-digital converter (ADC), consisting of a sample-hold amplifier (SHA), a voltage reference, and clock generation circuitry. Its data format is twos complement binary for bipolar mode. The input has a full-scale input voltage range of 10 V with a full-power bandwidth of 1 MHz, and can provide up to 128k conversions per second. This is just sufficient to sample a signal of 64 kcps.

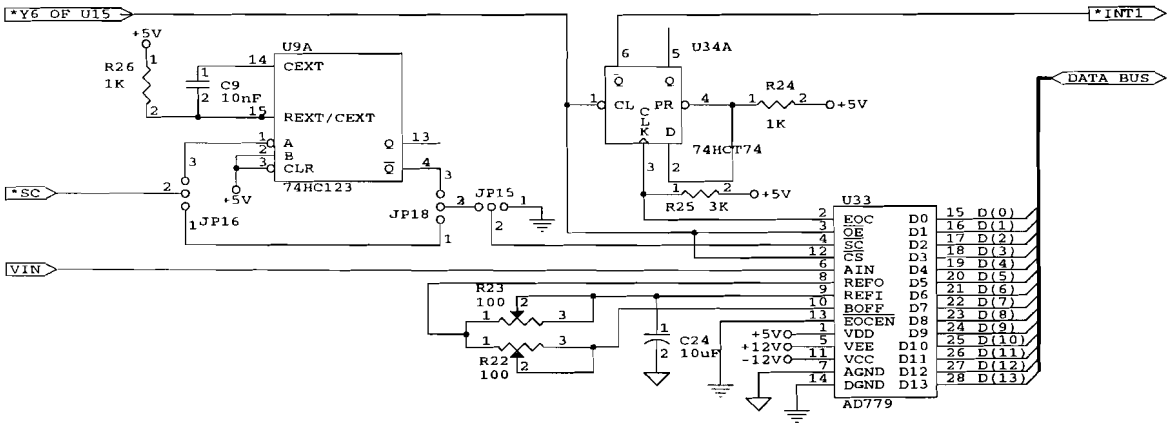


Figure 5.12: The analog-to-digital converter with peripherals.

Figure 5.12 shows the analog-to-digital converter with peripherals. In the transceiver circuit designed, the AD converter is used in the bipolar voltage range. Further specifications on the AD779 can be found in the data sheets. Two adjustable resistors, R22 and R23, are placed near the ADC to adjust offset voltage and gain, respectively. The calibration procedure can also be found in Appendix 2.

Figure 5.13 illustrates the timing of a conversion cycle. A conversion cycle is initiated by a high-to-low transition of the start-conversion input signal (*SC). After the conversion is started, the end-of-conversion signal (*EOC) goes low, indicating that the ADC is busy. After the signal is converted, the digital output becomes valid and *EOC goes high. The *EOC signal triggers a D-flip flop which generates an interrupt (*INT1) at the processor. In the interrupt routine, the digital output is read into the processor. Reading the output resets the flipflop and disables the interrupt. The conversion cycle repeats when *SC goes low again.

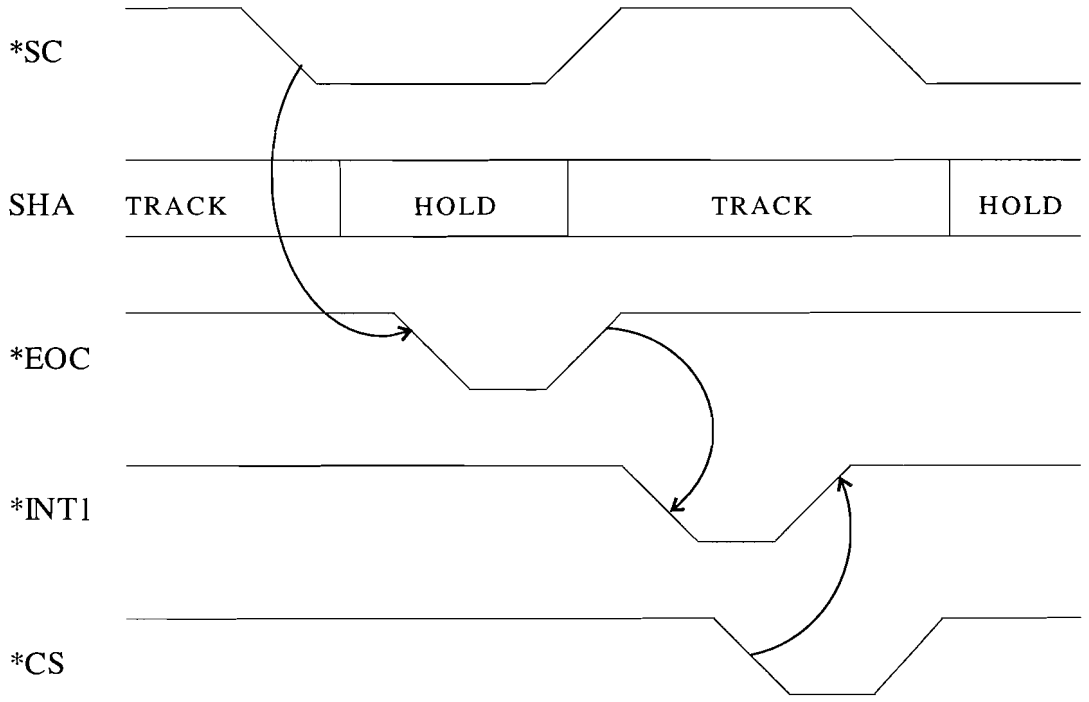


Figure 5.13: Analog-to-digital converter timing.

Figure 5.14 shows the ADC output timing. Because the maximum access time of the analog-to-digital converter is $t_{BA,\,max}$ = 100 ns, evaluating in a maximum access time of the ADC and selection circuit of $t_a(SL)$ = 107 ns, two wait-states are required.

The maximum output disable time of the analog-to-digital converter is $t_{FD,\,max}$ = 80 ns. This is far too long to avoid bus conflicts after a read operation of the ADC. A hardware solution of this problem would be to buffer its output by two 74F245 transceiver ICs. However, to save some space on the printed circuit board, a software solution has been implemented. If a read operation is followed by a NOP operation, the processor does not use the data bus during the next machine cycle. By doing this, the converter outputs are high impedant before the data bus is used again.

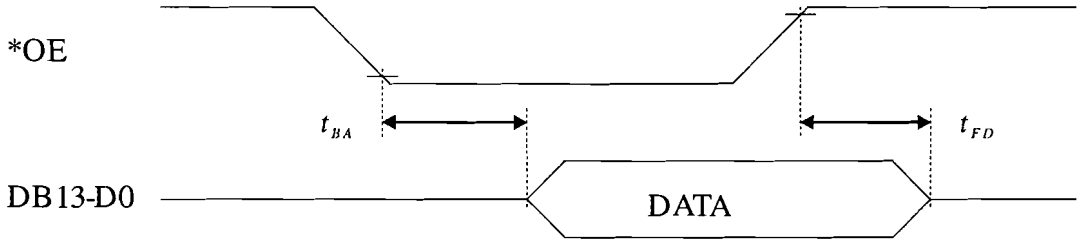


Figure 5.14: ADC-output timing.

The *SC signal is allowed to have a maximum low-time of 10 μs. If the *SC signal is a symmetrical square wave with a frequency below 50 kHz, the time the signal is low exceeds the maximum low-time of 10 μs. The circuit based on U9A shown in Figure 5.12 provides a signal with a maximum low-time of approximately 5 μs. By setting jumpers JP16, JP18, and JP15, the *SC signal can be connected directly to U33 or the *SC signal can be modified by U9A if the low-time is too long.

5.11 Transmitter configuration

A practical baseband direct-sequence spread-spectrum transmitter generates a specific pseudo-noise code and multiplies the binary data signal by it. Thus, it has a digital output which is afterwards modulated (e.g. BPSK). The transmitter designed can do a lot more. In software, not only the user's own code is generated, also pseudo-noise code sequences of other users can be generated, which are multiplied by random data signals. The spreaded data signal is superimposed on several other users' spreaded signals, thus, providing multi-user simulation. As discussed earlier, the output has to be analog.

The transmitter configuration is illustrated in Figure 5.15. The block 'TMS and peripherals' consists of digital signal processor, reset and oscillator circuit, wait-state generator, memory, I/O selection, and DIP-switches.

To generate a signal that consists of the transmitter spreaded data signal superimposed on the signals of several other users, the transmitter has an analog output. Thus, with one single transmitter multi-user simulation can be accomplished. To simulate an asynchronous system as well as possible, the code phases of the different signals should be random with respect to each other and in intervals much smaller than the chip period T_c. For this purpose, a high-speed DAC is used, which is discussed in Section 5.9.

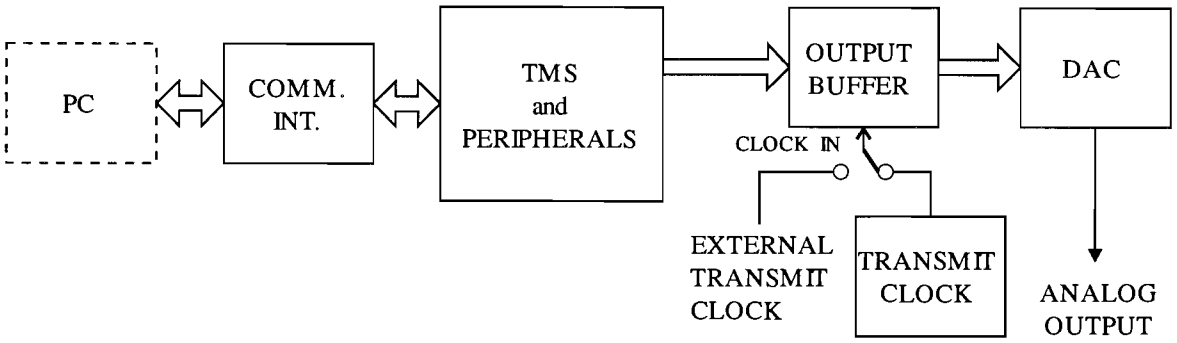


Figure 5.15: Transmitter configuration.

The chip rate (transmit clock rate) of the spreading code can be varied from 8 kcps to 64 kcps, and the smallest code phase difference is $T_c/16$, when $f_c = 1/T_c = 64$ kcps. This can be set by DIP-switches or programmed in software. The chip rate can also be controlled by an external clock, which will normally be synchronized with the modulator.

The external clock can also be used to test the tracking behaviour of asynchronous spread-spectrum systems, if Doppler shifts and oscillator instabilities influence the frequency of the received waveform. Further information on acquisition characteristics of the system designed can be found in the corresponding chapters about the theory and measurements.

The transmit procedure is as follows. After initialization, the transmitter circuit will read its DIP-switches and will get additional information and the data to be send from the personal computer. The pseudo-noise codes are generated and also the data signal, together with the random signals of all other users, are multiplied by the appropriate pseudo-noise codes, added, and stored in memory. As soon as the whole output signal is computed, these values are then send to the output buffer and clocked to the DAC. This procedure will be specified more accurately in Chapter 7 about the transmitter software.

5.12 Receiver configuration

In the receiver configuration, which is implemented on the same printed circuit board and illustrated in Figure 5.16, the transceiver is a soft-decision receiver. The analog input is sampled with a maximum frequency of 128 kHz, and converted to a 14-bit word. After conversion, an interrupt is generated and the 14-bit word is read into the processor where the sample is processed. The advantage of this configuration is an easy print design, but the disadvantage is that the processing is time-consuming, because each sample must be

processed separately. However, to test the behaviour of soft-decision spread-spectrum acquisition and to compare these characteristics with hard-decision results, this implementation is very useful.

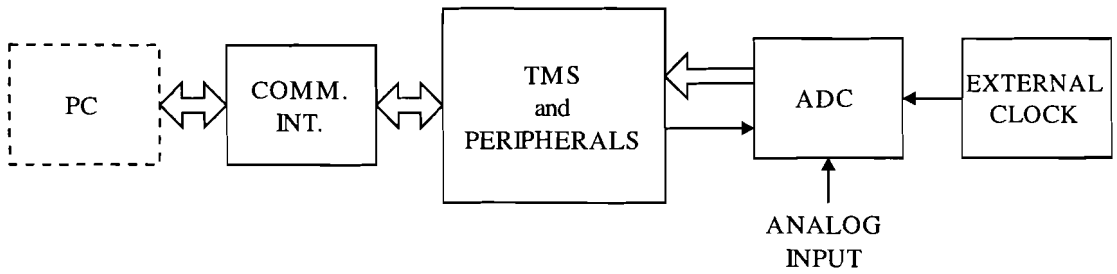


Figure 5.16: Receiver configuration.

Another solution, which can easily be implemented, is a hard-decision receiver. Each sample (one bit) is stored into a 16-bit shift register. After 16 samples are taken and the shift register is full, an interrupt is generated and the processor reads the contents of the shift register. Thus, 16 samples can be processed at the same time, which decreases the average processing time per sample, also because 16-bit words can be processed much more efficiently than single 14-bit samples. This decreases the average processing time per sample. The disadvantage of hard-decision is a 2 dB loss in signal-to-noise ratio after sampling with respect to soft-decision in 4 bits or more [9].

For implementation of the hard-decision receiver, the IC-socket of the 14-bit analog-to-digital converter is extended by another 2 pins to get full access to the 16-bit data bus. By plugging in a printed circuit board into the extended IC-socket, a hard-decision receiver can be easily build. The hard-decision circuit, that is not yet implemented, has full access to the data bus and uses the same selection and interrupt generation lines as the ADC.

The start-conversion signal (*SC) of the ADC or hard-decision circuit (HD circuit) is generated by an external frequency source. Tracking the code phase of the input spreaded waveform if frequency uncertainty is present, can be managed by controlling the frequency of the external frequency source. An example of an external frequency source is a frequency synthesizer consisting of a DAC, a voltage-controlled oscillator (VCO) and a pulse generator, as implemented in the circuit illustrated in Figure 5.17. To prevent the feedback loop to oscillate because of instabilities, the DAC has to be monotonic, i.e., the output increases or remains constant as the digital input increases. Since the DAC used in the transmitter configuration is fully monotonic over the temperature range, it can be used to control the VCO. Therefore, only a VCO and a pulse generator have to be provided for the implementation of a tracking system.

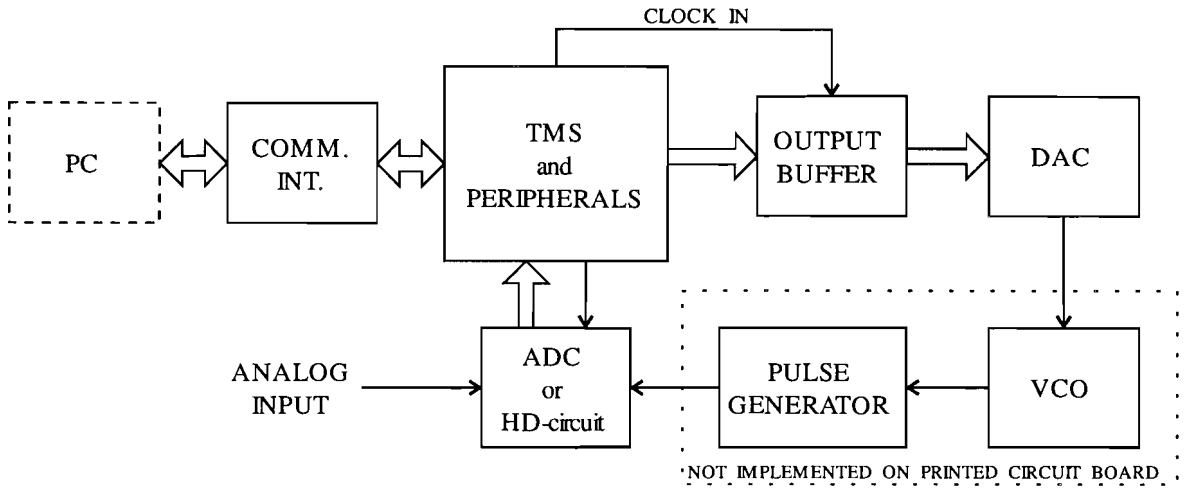


Figure 5.17: Receiver configuration with implemented frequency synthesiser.

The test procedure of the baseband spread-spectrum system can be subdivided into three phases. In the first phase, a synchronous system is made, using the internal transmit clock signal of the transmitter providing the start-conversion signal of the receiver. Thus, the acquisition behaviour of the receiver can be tested and the acquisition-time performance can be optimized. For lower chip rates, a transceiver can be configured both as a transmitter and as a receiver. The transmit clock signal of the transmitter can then serve as start-conversion signal of the receiver by setting jumper JP17. By connecting pins 1 and 2 of JP17, the internal transmit clock serves as *SC signal. When pins 2 and 3 are connected, an external *SC signal must be provided. In the second test phase, frequency uncertainty on the received waveform is generated by using an external transmit clock signal that has a small frequency uncertainty. The start-conversion signal of the receiver is held at a fixed rate. With this system, the acquisition-time behaviour in the presence of frequency uncertainty can be characterized. If acquisition has occurred, the maximum frequency uncertainty Δf_c at which the receiver stays in lock can be examined. If the frequency uncertainty is larger than Δf_c a tracking circuit is needed. In the final test phase, a fully asynchronous system with frequency uncertainty on the received waveform and a receiver with variable sample rate is made. The acquisition and tracking behaviour can be fully tested and optimized.

5.13 Print layout and EMC considerations

Great care should be taken for positioning of components on the printed circuit board (PCB) and routing of power-supply lines, ground lines, and high-speed lines (data bus, address bus, etc.). Components or connections can cause interference to each other and to

other printed circuit boards. This should be kept to a minimum. In [10] about printed circuit board design, some useful rules on positioning of components and routing are listed. The most important ones are listed below and applied as much as possible, hopefully resulting in a better performance and less radiation of the printed circuit board. The print design is shown in Appendix 1.

General rules on PCB design:

* keep digital and analog circuitry separated;
* high-speed logic should be kept together in functional blocks;
* memory circuits should be located far away from I/O circuits and their runs preferably near the centre of the PCB;
* oscillators and clock circuits should be located near the centre of the PCB. This minimizes coupling to I/O runs and places the higher-frequency sources in the centre of the power and ground distribution system;
* initiate placing the oscillator circuit, processor, memory, and fast I/O;
* keep connections as short as possible;
* isolated digital and analog power supplies should be used when mixing digital and analog circuitry on a PCB. Single-point common grounding of the analog and digital power supplies should be performed at one point and one point only;
* ground power runs should be routed first on the PCB, with power runs routed parallel to ground runs;
* use as many ground loops as possible on the PCB;
* add decoupling capacitors. These should be placed as close as possible to the power pins of the integrated circuits. High-frequency decoupling should be done with ceramic capacitors since the inductance of the capacitor is less than with other types;
* after design 'landfilling' is carried out. This includes widening the ground grid runs where possible;
* connect the PCB's ground systems to the metal chassis of the unit; The chassis will serve as a shield.

Some supplementary rules on analog circuitry:

* keep related parts very close together and connections very short;
* circuits must be separated to keep unwanted interaction among them to a minimum;
* the surrounding space can be floated with ground, thus improving the ground plane distribution and further isolating the circuits;
* add decoupling capacitors.

Decoupling capacitors are placed for two purposes. They supply transient current to the chip during gate switching. Secondly, they limit the size of a potential radiating loop associated with the power supply to a switching gate. To minimize this radiation and to save some space on the PCB, special IC-sockets with decoupling capacitors assembled in it are provided.

5.14 Hardware errors

The following hardware errors have occurred until now:

- for disabling the interrupt signal for *INT1, it is not taken into account that the *IACK signal is undefined when CLKOUT1 is high. Because of the very fast TTL logic, the interrupt signal can be disabled before it is recognized by the signal processor. This problem is solved by using a much slower and port (74LS08) instead of the one initially chosen (74F08);
- the enable signal of multiplexer U28 was initially connected to +5V. By taking the IC-pin out of the socket and connecting it to digital ground, the multiplexer is always enabled;
- the transceiver does not function if there is one wait-state set for I/O (not parallel port I/O). The reason for this is unknown.
- a hardware error concerning the communications-interface hardware is discussed in Section 6.2.2, on page 62.

5.15 Conclusions and recommendations

The electronic circuit designed can be used to test many characteristics of an asynchronous direct-sequence spread-spectrum system. In a test procedure, which is divided into three phases, the acquisition and tracking behaviour of the system can be tested step by step. Soft-decision as well as hard-decision detection could be performed and characterized.

However, the circuit is an all-round transceiver that can be used for a lot more purposes. Almost all relevant signals are available at the 64-pins connector, including programmable test signals. The ADC has a large bandwidth which makes it suitable for sub-sampling, and the maximum sample frequency of 128 kHz is large enough for many applications. The high-speed DAC and matching operational amplifier are also applicable for many purposes, e.g., for signal generation. All this provides a circuit that is useful for many applications involving signal processing.

Apparently, the EMC considerations described in Section 5.13 have been contributing to the good operation of the electronic circuit. All the control signals as well as the signals on the data and address buses are almost free of interference. Also the analog input and output signals have acceptable noise levels. All this confirms that the time investment in a good print layout is well spent.

There are a few items the user/programmer of the hardware has to take care of. They are listed below:

- it is possible to write to the EPROM transceivers. This causes a data-bus conflict that can damage the transceiver ICs, but more probably the digital signal processor;
- reading the analog-to-digital converter is only allowed if the read instruction is executed from the on-chip RAM and is followed by a NOP instruction. Otherwise, a bus conflict can occur that can damage the hardware;
- the analog-to-digital converter is now connected to the 14 least significant bits of the data bus; in future designs it is better to connect the ADC to the 14 most significant bits, because the sign bit of the ADC (bit 13) is then also the sign bit of the input word that the processor reads into memory;
- to prevent the OpAmp connected to the output of the digital-to-analog converter to oscillate due to capacitive load of connected coaxial cables, a resistor of 39 Ω is connected from the output of the OpAmp to the output BNC plug and the feedback loop; this specific resistance also makes the output impedance of the OpAmp equal to the impedance of the coaxial cables connected.

6 The communications interface

6.1 Introduction

The spread-spectrum transceiver has an interface to communicate with a personal computer (PC). The communications interface has already been described in [11], but hardware as well as software concerning control of communications by the personal computer have not been described. Because these aspects form the basics of the design, the communications interface is described again completely in this chapter. However, considerations concerning the choice of this specific interface can be found in [11].

The communications interface was designed to provide communications between a personal computer and up to eight devices. Because a baseband picoterminal system consists of one baseband transmitter and one baseband receiver, only two devices have to be connected to the personal computer. This chapter, however, gives a general description of the communications interface. The baseband transmitter and receiver are referred to as devices.

6.2 Communications-interface hardware

6.2.1 The communications interface on the personal computer

The communications interface is based on the Centronics parallel interface standard [12], best known for its printer interface purpose. The Centronics port on a personal computer consists of 12 digital outputs and 5 digital inputs. A 25-pins female connector is the standard Centronics connector on a personal computer. Figure 6.1 shows the connector front view. Table 6.1 shows the standard Centronics interface pinout.

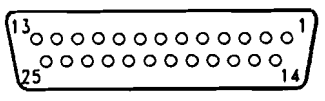


Figure 6.1: Front view of a female Centronics connector.

Table 6.1: Standard Centronics interface pinout.

Pin nr.	I/O	Function	Pin nr.	I/O	Function
1	O	Strobe	10	I	Acknowledge
2	O	Data 0	11	I	Busy
3	O	Data 1	12	I	Paper Out
4	O	Data 2	13	I	Select
5	O	Data 3	14	O	Auto Linefeed
6	O	Data 4	15	I	Error
7	O	Data 5	16	O	Reset
8	O	Data 6	17	O	Select In
9	O	Data 7	18-25		Ground

The communications bus between PC and transceiver uses all 17 available I/O lines. One input is used to generate an interrupt on the personal computer, as the other 4 inputs are used for data reception. Of the 12 outputs, one is used for interrupt generation on a device and 8 are used for data transmission to a connected device. Thus, 3 output lines are left to select one out of a maximum of 8 devices connected to the communications bus. Table 6.2 shows the communications interface pinout as implemented on the PC. Figure 6.2 shows a schematic diagram of the communications bus implemented.

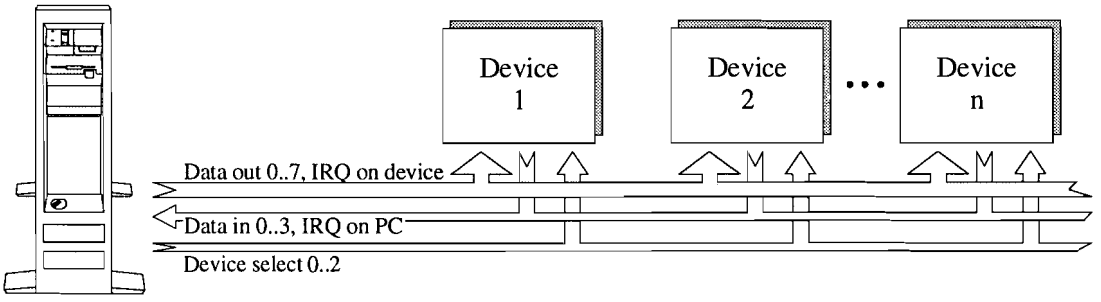


Figure 6.2: The communications bus implemented.

The ROM-BIOS of a PC can support up to three Centronics ports. A Centronics port is mapped to three 8-bit internal I/O ports, which can be programmed or read by the personal computer. The I/O-port addresses are referred to as LPT, LPT+1, and LPT+2. During initialization, the Centronics ports are tested in the following order (the port addresses refer to the first I/O-port addresses, i.e. LPT; the $-sign indicates a hexadecimal number):

Table 6.2: The communications-interface pinout as implemented on the PC.

Pin nr.	I/O	Function	Pin nr.	I/O	Function
1	O	IRQ on device	10	I	IRQ on PC
2	O	Data out 0	11	I	Data in 3
3	O	Data out 1	12	I	Data in 2
4	O	Data out 2	13	I	Data in 1
5	O	Data out 3	14	O	Device select 0
6	O	Data out 4	15	I	Data in 0
7	O	Data out 5	16	O	Device select 1
8	O	Data out 6	17	O	Device select 2
9	O	Data out 7	18-25		Ground

1 port $3BC, found on Monochrome Display/Printer Adapter (MDPA)
2 port $378, printer port adapter #1
3 port $278, printer port adapter #2

The BIOS assigns the Centronics ports as LPT1 to LPT3, as they are discovered. It stores the port addresses that respond to the test starting at memory place $0:$0408 in the PC.

Table 6.3 gives an overview of I/O ports of the Centronics interface. The address LPT refers to the port address mentioned above, thus, LPT = $3BC for the MDPA, LPT = $378 for the parallel printer adapter #1, and LPT = $278 for the parallel printer adapter #2.

The Centronics port is interrupt driven. The hardware interrupt is enabled by setting bit 4 of LPT+2. An interrupt is generated when Acknowledge (pin nr. 10) goes low. For LPT1 the interrupt number is $0F. For LPT2 the interrupt number is $0D. The interrupt service routine sends new data to the output ports and provides the control signals.

6.2.2 The communications-interface hardware on a device

The interface hardware on a device is shown in Figure 6.3. The interface has been designed so that the personal computer can exchange data with one up to a maximum of eight devices. Each device has a bus address specified by the DIP-switches 8 to 6 of S2 (DIP-switch 8 is the least significant one). The device is selected if the address set by the DIP-switches is the same as the one addressed by the personal computer via inputs IN8 to

Table 6.3: Overview of the I/O ports for the Centronics interface.

Port address	Description	Bit	Pin number
LPT	Printer Data Latch Write: send byte to printer Read: fetch last byte sent		
	Data 0 to Data 7	0 to 7	2 to 9
LPT+1	Printer Status Read-only		
	Error	3	15
	Select	4	13
	Paper Out	5	12
	Acknowledge	6	10
	Busy	7	11
LPT+2	Printer Controls Read/Write		
	Strobe	0	1
	Auto Linefeed	1	14
	Reset	2	16
	Select In	3	17
	IRQ Enable	4	-

IN10. Communication is only possible between the personal computer and the device selected. Input IN11 on the device is connected to ground. This enables the user to disable a device permanently by opening DIP-switch 5 of S2.

Communication is fully interrupt driven on both the personal computer as well as on a device. Both the personal computer and a device can initiate communication. However, a device must be selected by the personal computer to be able to start communication. In practice this means, that a device can only start communication if the PC expects the device to start communication, and thus if the device is selected.

If the personal computer starts communication, a positive edge of 'IRQ on device' on U20A generates an interrupt (*INT2) on the digital signal processor of the device, and, at the same time, data are put onto the bus. The interrupt causes the device to start execution of the interrupt service routine, in which the digital signal processor of the device reads

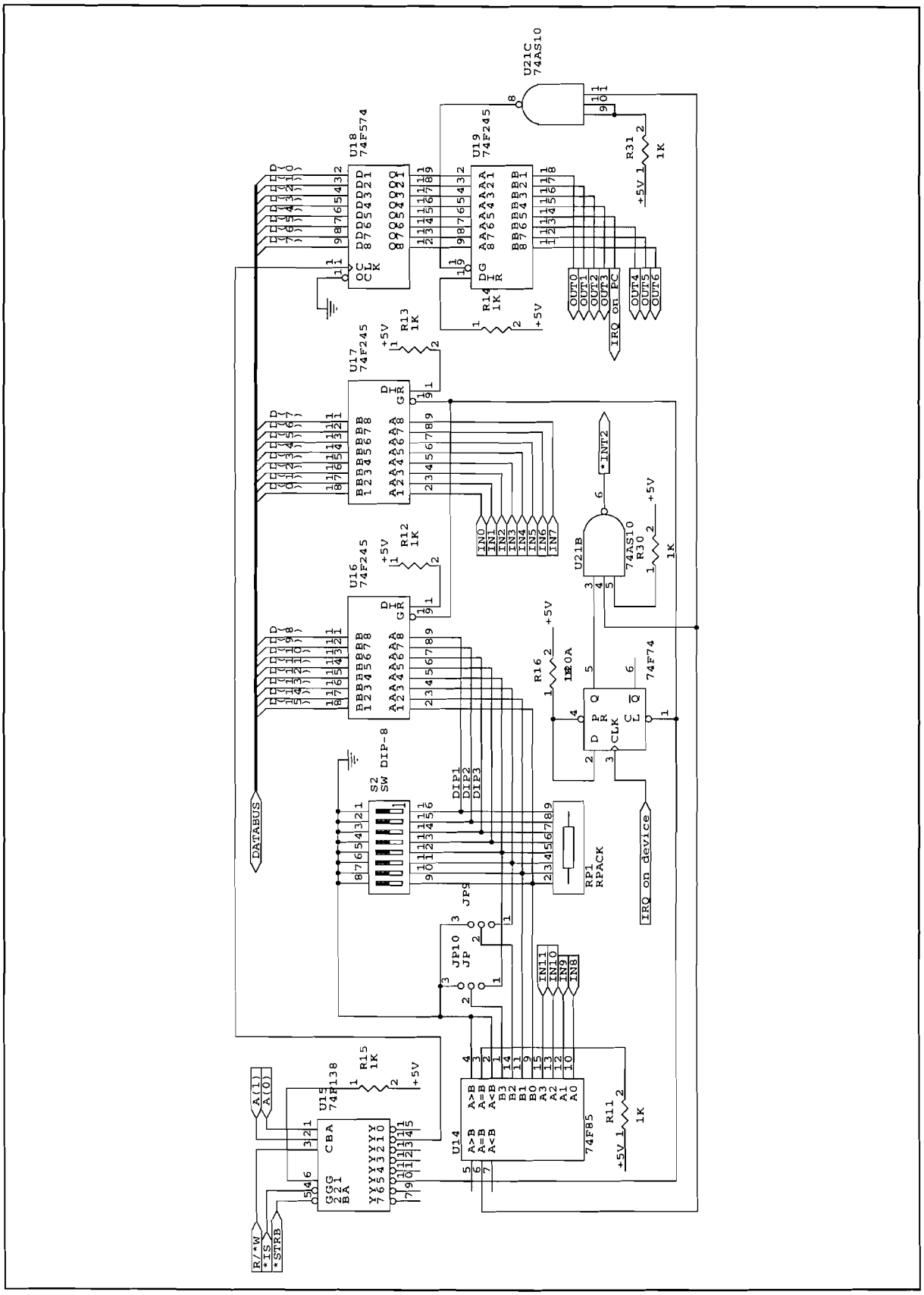


Figure 6.3: The communications-interface hardware.

the data from the PC into its memory. The inputs are driven by transceivers ICs U16 and U17, which are enabled by output *Y5 of U15. The principle and timing of the read operation are almost the same as for a memory-read operation as illustrated in Figure 5.8. The only difference with a memory-read operation is that the processor uses two wait-states instead of zero to read the data into memory.

If communication is initiated by a device, a 4-bit word is clocked into buffer U18 and a hardware interrupt on the personal computer is generated by a positive edge of 'IRQ on PC'. The buffer is driven by *Y1 of U15. If the device is selected, data is put onto the bus by transceiver IC U19, which is enabled by the output at pin 6 of comparator U14. If a device is not selected, the outputs of U19 are high impedant and both the personal computer as well as the device itself are unable to generate an interrupt.

Although the personal computer cannot generate an interrupt on a device that is not selected, the D-flip flop of an unselected device is set though. Because selection of a device is fully done by hardware, the device cannot reset the flip flop in case the device is not selected. Thus, if the personal computer changes the address via the three device-select lines, the flip flop provides an unwanted interrupt on the digital signal processor. This hardware design error can easily be corrected by changing the order of the flip flop U20A and the NAND port U21B. In this manner the flip flop providing an interrupt is only set if a device is selected.

Because a baseband picoterminal only consists of one baseband transmitter and one baseband receiver, only two devices must be connected to the communications bus for this application. The transmitter is addressed as device 0, while the receiver address is 1. Jumpers JP9 and JP10 can be used to connect two inputs of comparator U14 to ground. This enables the programmer of a device to use DIP-switches 5 and 6 for other purposes.

6.3 The communications interface protocol [11]

6.3.1 Introduction

The different hardware communication standards do not automatically include a communication protocol. Often data are transmitted and assumed to be received correctly (e.g. computer to printer). In other applications, such as file transfer via a modem, data are transmitted in packets and hardware or software checks for possible errors are done. If an error is detected, data are corrected or retransmitted.

The communications protocol used here is based on the X-modem file transfer protocol. Initially, packets, which are comparable with the ones specified for X-modem, contain transmission data and a cyclic redundancy check (CRC) for error detection. Because communications appeared to be error-free, the cyclic redundancy check is not implemented in software here. The communications protocol and software with the cyclic redundancy check implemented are discussed in [11].

6.3.2 The communications protocol

Communication is fully interrupt driven to minimize processor occupation. The data is sent in packets, consisting of a header, the data, and a packet ending. Data transmitted by the personal computer are sent in bytes, while data transmitted by a device are sent in 4-bit words.

If communication is initiated by the personal computer (respectively a device selected by the personal computer), it interrupts the program on the device selected (respectively the PC) by sending a character and continues its own task. As soon as the device (respectively the PC) has processed the received data, it sends an acknowledge and continues the interrupted program. The running program on the personal computer (respectively the device selected) is interrupted and the acknowledge is received. It sends the next character of the message if the packet has not been completely transmitted yet, which is again answered with an acknowledge. Timers on both the PC and the device check the occurrence of time-outs.

A communication packet transmitted from the personal computer has the following structure:

SOH	LEN	D_1	D_2	...	D_{LEN-2}	EOT

Every character transmitted from the personal computer is 8-bit wide. The meaning of the packet fields are:

SOH	start of header;
LEN	length of packet in bytes with a maximum of 255 (excluding EOT);
$D_1..D_{LEN-2}$	data fields (all ASCII characters);
EOT	end of transmission.

A communication packet transmitted from a device to the personal computer has the following structure:

SOH	LEN_2	LEN_1	LEN_0	$D_{1,h}$	$D_{1,l}$	$D_{2,h}$	$D_{2,l}$	...	$D_{LEN-4,h}$	$D_{LEN-4,l}$	EOT

Every character transmitted from a device is 4-bit wide. The meaning of the packet fields are:

SOH	start of header;
LEN_0..LEN_2	length of packet in nibbles with a maximum of 4095 (excluding EOT). Length = $256*LEN_2 + 16*LEN_1 + LEN_0$;
$D_{1,h}$..$D_{LEN-4,l}$	data fields; data bytes are formed by $16*D_{n,h} + D_{n,l}$;
EOT	end of transmission.

The acknowledge procedure is as follows:

- the acknowledge on a SOH is an attention-acknowledge (AACK);
- the acknowledge on all characters but the first and the last is a data-acknowledge (DACK);
- the acknowledge on the last character is an EOT. An EOT is acknowledged by an EOT from the transmitting side;
- if a DACK or an EOT is expected but not received, it is acknowledged by a negative-acknowledge (NACK), and the communication procedure is started again.

In case of a baseband picoterminal system, only two devices, i.e. a baseband transmitter and a baseband receiver, are connected to the communications bus. In first instance, the software for both the PC and the transceiver was written so that only the PC could initiate communication and the transceiver could only transmit an answer to the message or data received from the PC. For the measurement system that is described in Chapter 8, it is necessary that the baseband receiver can also initiate communication. Therefore, the communication-software routines were extended.

Figure 6.4 illustrates the communications protocol implemented. Note that not all possible occurrences of errors are shown. Two errors that can occur but are not illustrated are that a DACK or an AACK was expected but not received. If this happens, a NACK is sent and the communications procedure is started again. In Figure 6.4, the protocol is shown for communication initiated by the personal computer. However, the protocol is the same for communication initiated by a device.

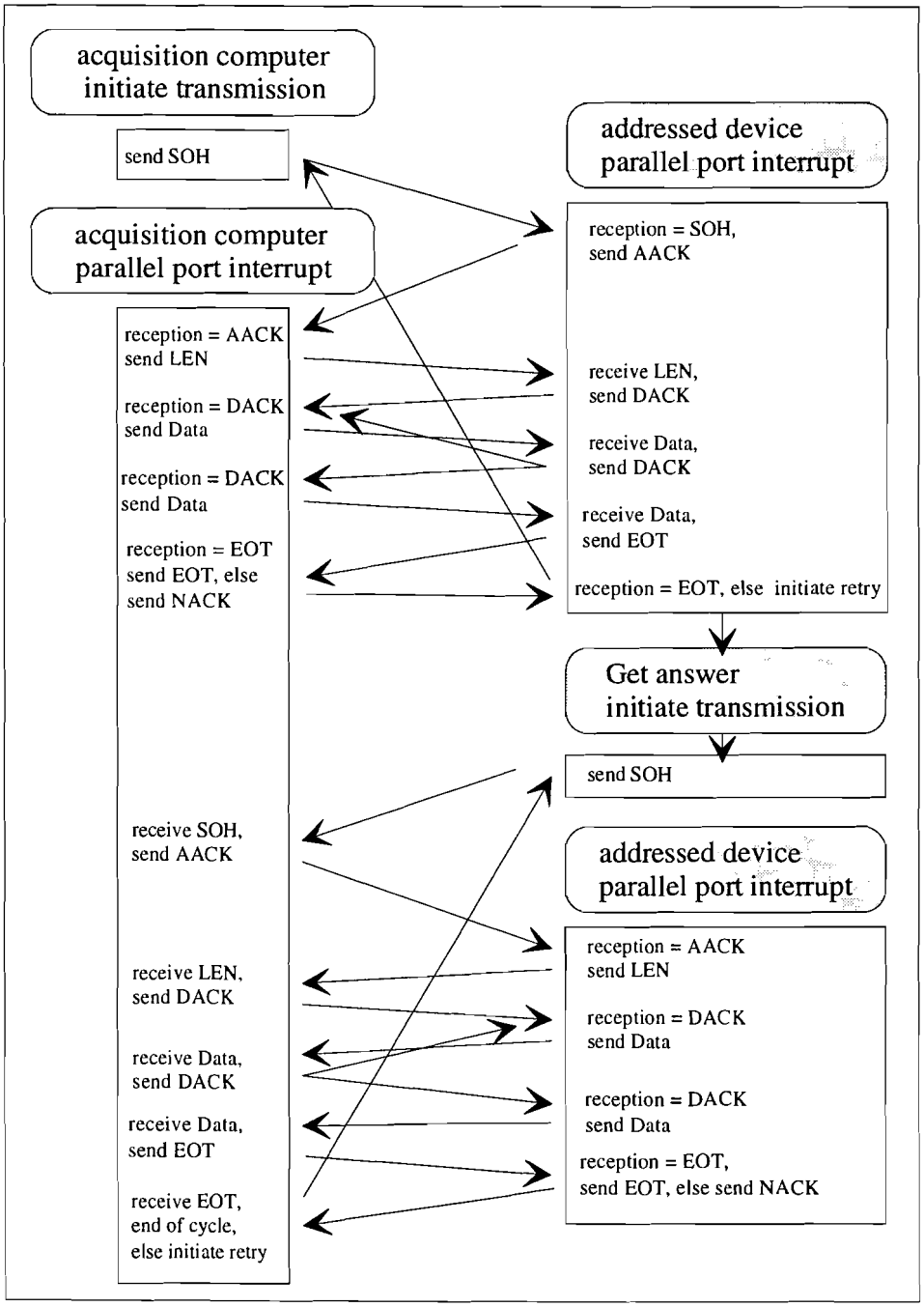


Figure 6.4: The communications protocol implemented.

6.4 The communications-interface software

6.4.1 Introduction

The communication routines form an almost independent part of the software. The communications protocol is designed in such a way that if a communication cycle is started, all further actions are interrupt driven and do not require attention of the main application. From this point, the device software discussed is specifically for the transceiver implemented. A more general discussion can be found in [11].

6.4.2 Transceiver and PC software

Figure 6.5 shows the flow chart of the parallel-port interrupt routine. The communications protocol is implemented in both the transceiver interrupt routine as well as in the parallel-port interrupt on the personal computer. The global variable PAR_Status indicates the status of communications. The variable may have the following values:

0 No active communication;
1 The PC is transmitting a command or data to a device;
2 The PC has transmitted a command or data to a device, which is processing the received command or data and is generating an answer;
3 An answer or data of a device is being transmitted to the PC;
4 A fatal error occurred and communications were terminated.

The interrupt routine on the transceiver is invoked by hardware interrupt 2. Only a small part of the communication software was implemented in the main program. This part concerns the recognition of commands, and monitoring of PAR_Status. The interrupt on the personal computer is invoked by hardware interrupt $0D or $0F, depending on the printer port. The personal computer monitors the variable PAR_Status in the main pogram.

If communication is started by the personal computer and a message has been received by the transceiver, the variable PAR_Status becomes 2. This variable is monitored in the transceiver main-program loop and a routine is started to get an answer to the received message. After the answer is found, the first character of the answer is transmitted to the personal computer. After transmission of this first character, communication is again fully interrupt driven. The timer interrupt on the transceiver provides generation of the interrupt signal on the personal computer and detection of timeouts.

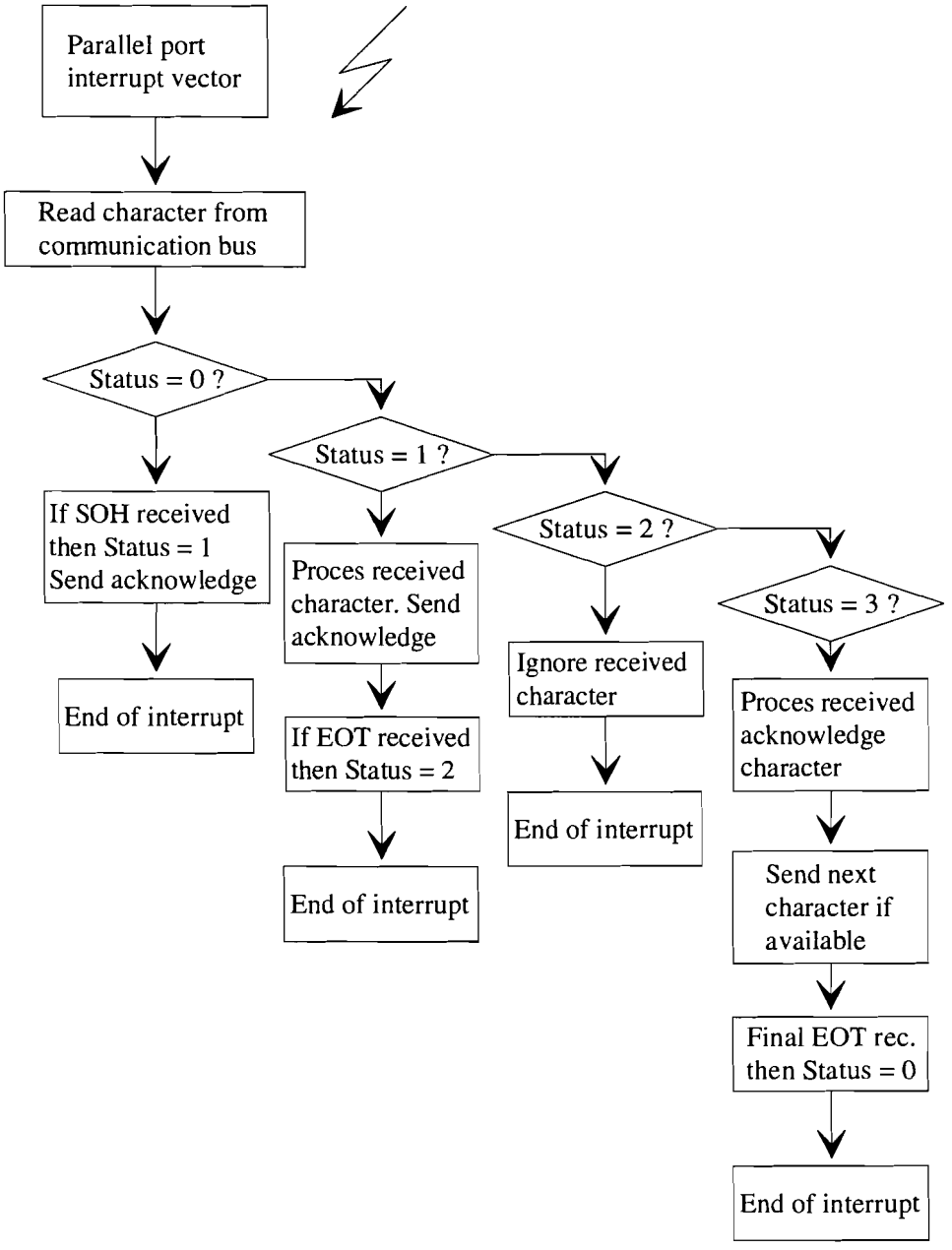


Figure 6.5: Flowchart of the parallel-port interrupt.

At the end of a measurement (see Chapter 8 for the measurement procedure), the baseband receiver initiates communication. The procedure is equal to the answering procedure of a transceiver as discussed above. After the measurement data are sent, communication is ended. Of course, the receiver is only able to start communication on the condition that it is selected by the personal computer. This is implemented in the personal-computer software.

Commands for the transceiver can be implemented in software very easily. This is done to adjust the software to the application of the device. The data structure specifying the communication commands consist of three sections:

- one word specifying the number of commands defined;
- pointers to the command routines and command-string specifications;
- one word specifying the routine address belonging to the command string immediately followed by a word specifying the length of the command string in characters and the command string itself.

Figure 6.6 illustrates the applied data structure. With this structure, new commands can be implemented and commands can be easily changed without changing the communication routines.

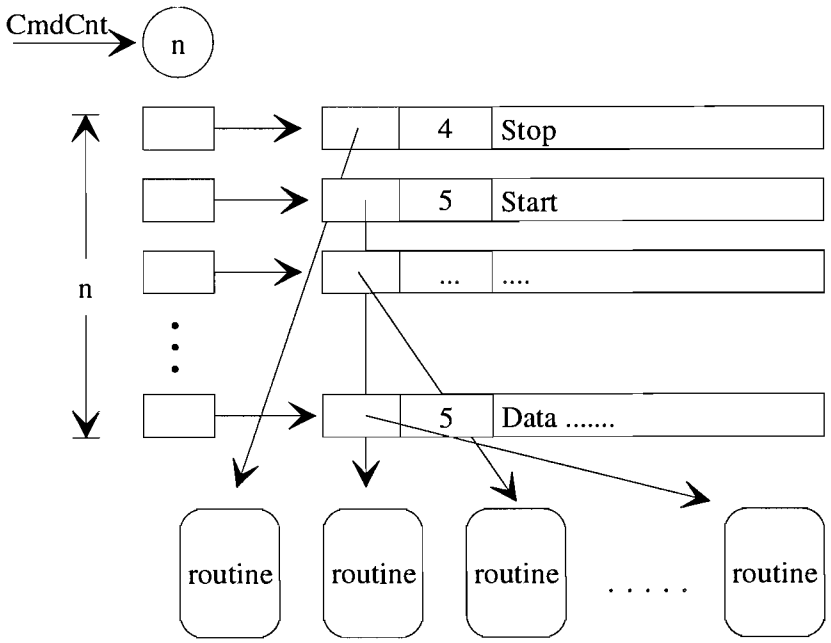


Figure 6.6: Data structure for parallel-bus communications.

The assembly source of the transceiver program is listed in Appendix 4. The program source listing of the personal-computer program can be found in Appendix 5. Further details and comments on the implemented routines are mentioned in the source listings.

7 Software

7.1 Introduction

This chapter describes the transceiver and personal-computer software structure. The transceiver software was written in TMS320C25 assembly language using the Texas Instruments Software Development System (SWDS). Details on the SWDS and its accessory software such as the assembler and linker program can be found in Appendix 3. The personal-computer software was written in the Turbo Pascal 7.0 programming language.

7.2 Transceiver software

7.2.1 Introduction

The transceiver can be configured as a transmitter or as a receiver. The software for both the transmitter and the receiver configuration is integrated in one and the same program. It enables the user to perform measurements with the acquisition system designed. The measurement procedure is discussed in Chapter 8. After initialization, program execution for the transmitter and the receiver is fully interrupt driven. Only a small loop in the main program will check some variables concerning the status of the transmitter and receiver. In the next sections, the overall program structure will be explained, and flow charts are shown of the main program, the transmit interrupt and the receive interrupt. The interrupt routine concerning parallel port communications has already been discussed in Chapter 6. Program details are further explained in the assembly source which is listed in Appendix 4.

7.2.2 Initialization and main loop

Figure 7.1 illustrates the flow chart of the transceiver main program. During initialisation the program copies itself from EPROM to RAM. The processor does not use wait-states to access the RAM as it does to access the EPROMs. Program excution from RAM enables the processor to execute the program faster. Details on memory access are discussed in Section 5.7. After the program has been copied into RAM, variables are set and the processor reads the DIP-switches. Depending on the status of DIP-switch 8, the transceiver

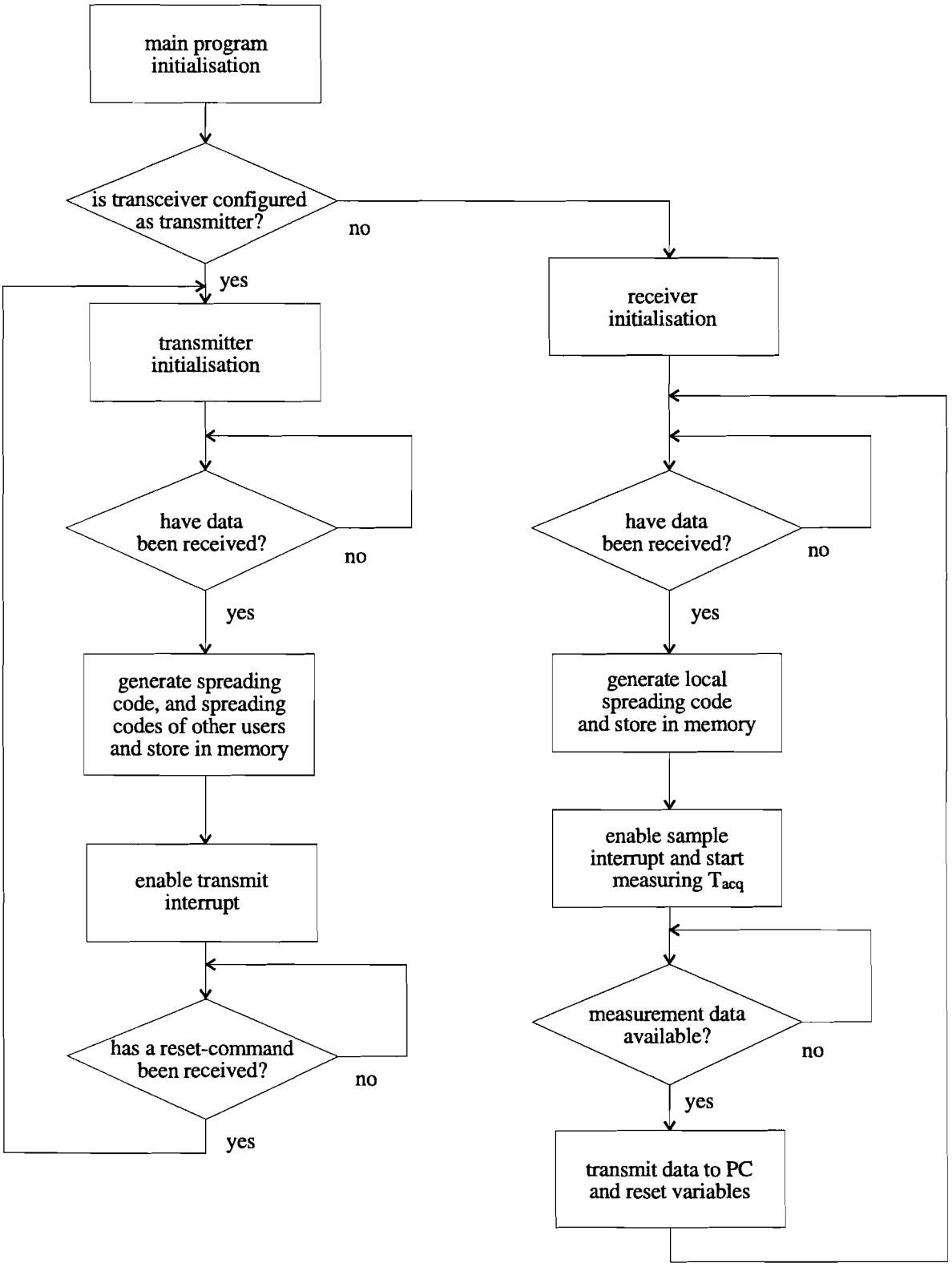


Figure 7.1: Flow chart of the tranceiver program.

is configured as a transmitter or as a receiver. If DIP-switch 8 is closed, the transceiver is configured as a receiver, otherwise as a transmitter. Note that DIP-switch 8 also sets the device address for communication with a personal computer.

If the transceiver is configured as a transmitter, a small transmitter initialisation is performed and the transmitter waits for data from the PC. When data has been received, the transmitter starts generating its spreading code and the spreading codes for a number of other users. These are stored into memory. The baseband spread-spectrum signal is clocked out to the DAC on interrupt basis. After a measurement is performed and a reset command has been received from the PC, the transmitter resets all variables and waits for new data.

If the transceiver is configured as a receiver, a small receiver initialisation is performed after the main initialisation, and the receiver waits for data from the PC. If data has been received, the local spreading code is generated and stored into memory and the receive interrupt is enabled. In the receive interrupt, which is discussed in Section 7.2.4, a correlation of the input signal and the local spreading code is calculated. The main program monitors if the correct code phase has been acquired and measurement data are available. When these data are available, they are sent to the personal computer and the receiver resets all variables and waits for new data. After the PC has written the measurement data to file, a new measurement can be performed.

More details on initialisation, possible commands, the data sent to transmitter and receiver, and the generation of spreading codes can be found in the assembly source listing in Appendix 4.

7.2.3 Transmit interrupt routine

Figure 7.2 illustrates the flow chart of the transmit-interrupt routine. The transmit interrupt routine is invoked by hardware interrupt 0, which is generated by the transmit-clock signal. In the interrupt service routine, the transmitter reads a chip of its spreading code and the chips of the spreading codes of the other users from memory. The data signal, that is sent by the PC, is modulated by the chip of the own spreading code and random signals are modulated with chips of the other spreading codes. These signals are added, scaled and written to the digital-to-analog converter.

The chip of the own spreading code and chips of the spreading codes of the other users are indicated by counters. At the end of each code-length period (after N chips have been sent), these counters are reset.

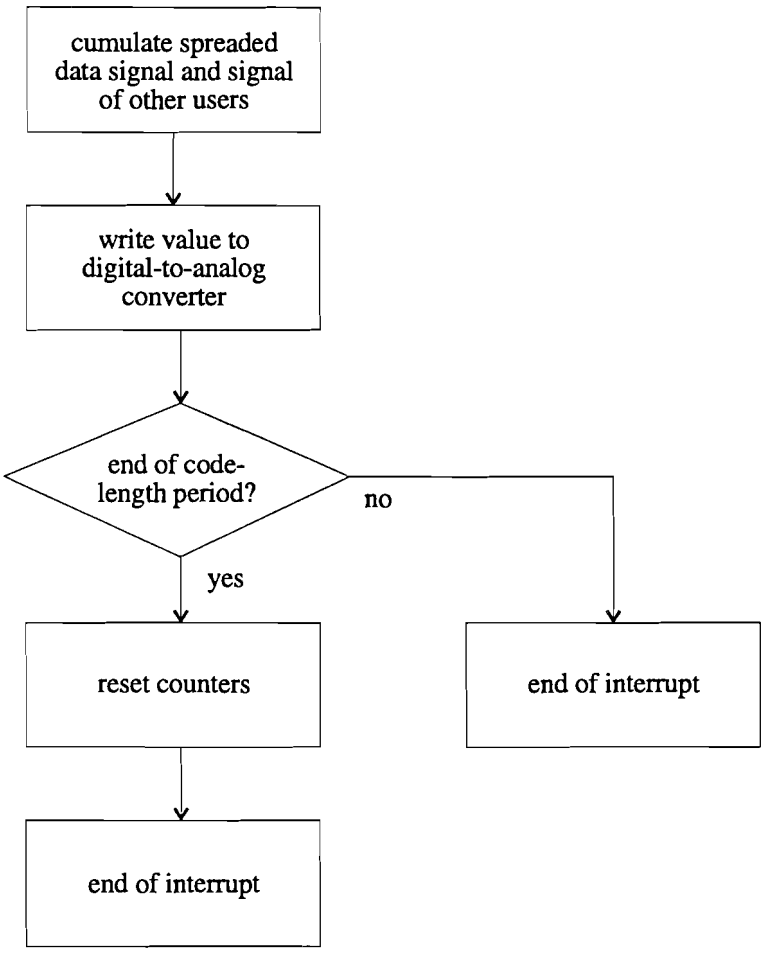


Figure 7.2: Flow chart of the transmit-interrupt routine.

7.2.4 Receive interrupt routine

Figure 7.3 shows the flow chart of the receive-interrupt routine. The interrupt service routine is invoked by hardware interrupt 1. The receive-interrupt service routine performs a correlation of the input signal and the local spreading code. A sample of the input signal is multiplied with a chip of the local spreading code. The result is accumulated to calculate the correlation. At the end of each dwell time, the calculated correlation is compared with a threshold. If the correlation does not exceed this threshold, the code phase of the local spreading code is updated and a new correlation is calculated. If the threshold is exceeded, the receiver goes into a verification mode and the number of successive threshold exceedings in counted. When the threshold is exceeded for three successive times, acquisition is declared. A variable is set, which indicates the main program that measurement data is available. This causes the receiver to send the measurement data to the personal computer. This was already explained in Section 7.2.2.

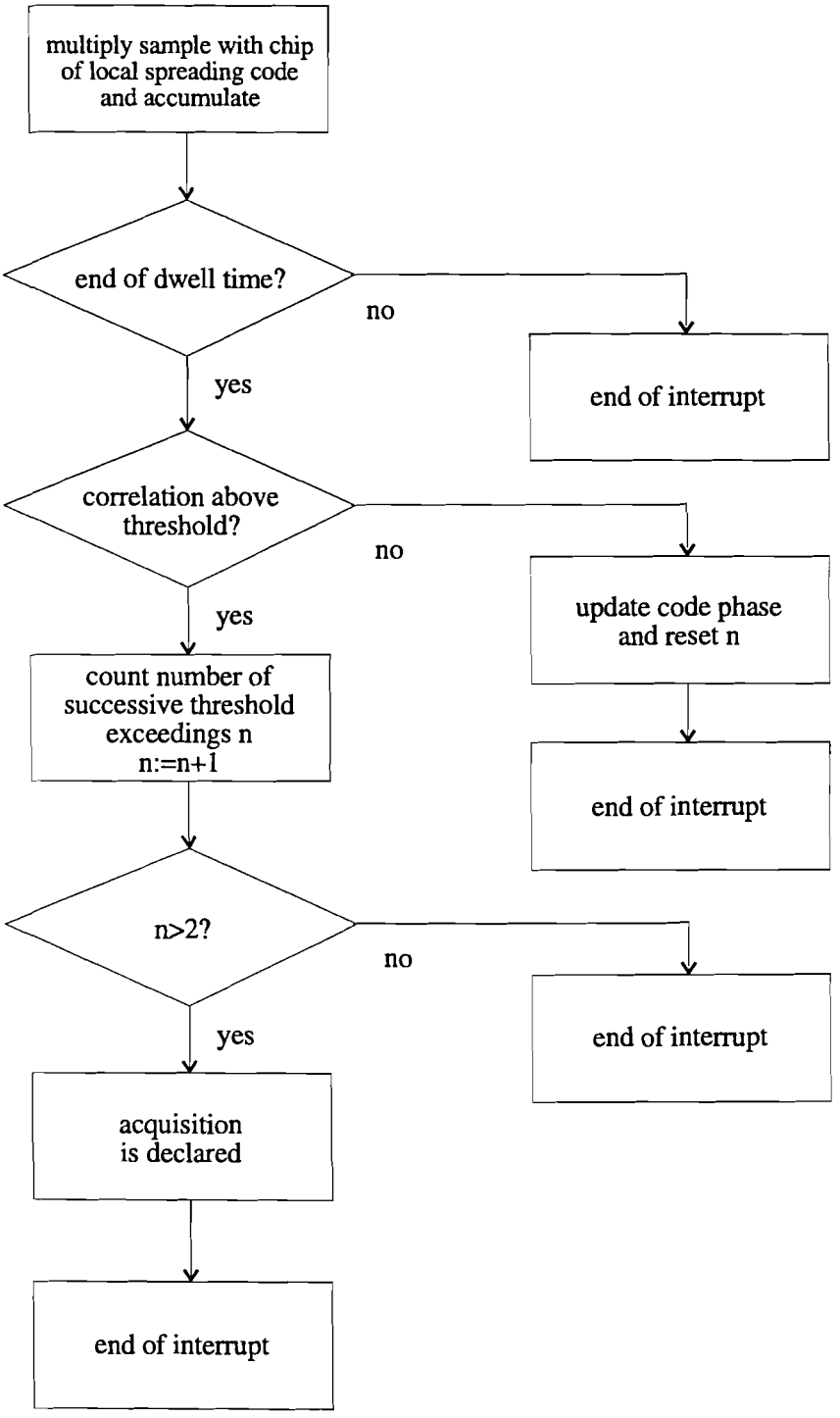


Figure 7.3: Flow chart of the receive-interrupt routine.

7.3 Personal-computer software

The personal computer enables the user of the measurement system described in Chapter 8 to provide the transmitter and receiver of the appropriate data and to store measurement data. The measurement procedure discussed in Section 8.3 is implemented in the personal-computer program. Figure 7.4 shows the flow chart of the personal-computer program. It is written in Turbo Pascal 7.0 and the program source is listed in Appendix 5.
After initialization, the program reads the measurement parameters from file. These are sent to the transmitter and to the receiver, respectively. Then, the personal computer waits until the receiver has acquired the correct code phase and returns the measurement data. These data are saved to file and both the transmitter and the receiver are reset. If more measurements have to be performed, again measurement parameters are read from file and the whole procedure repeats. Otherwise the program is ended.

The PC program source is listed in Appendix 5. Details are explained in the comment lines of the program source listing. Details on interrupt handling in a personal computer can be found in [13] and [14]. Details on the Turbo Pascal programming language can be found in [15].

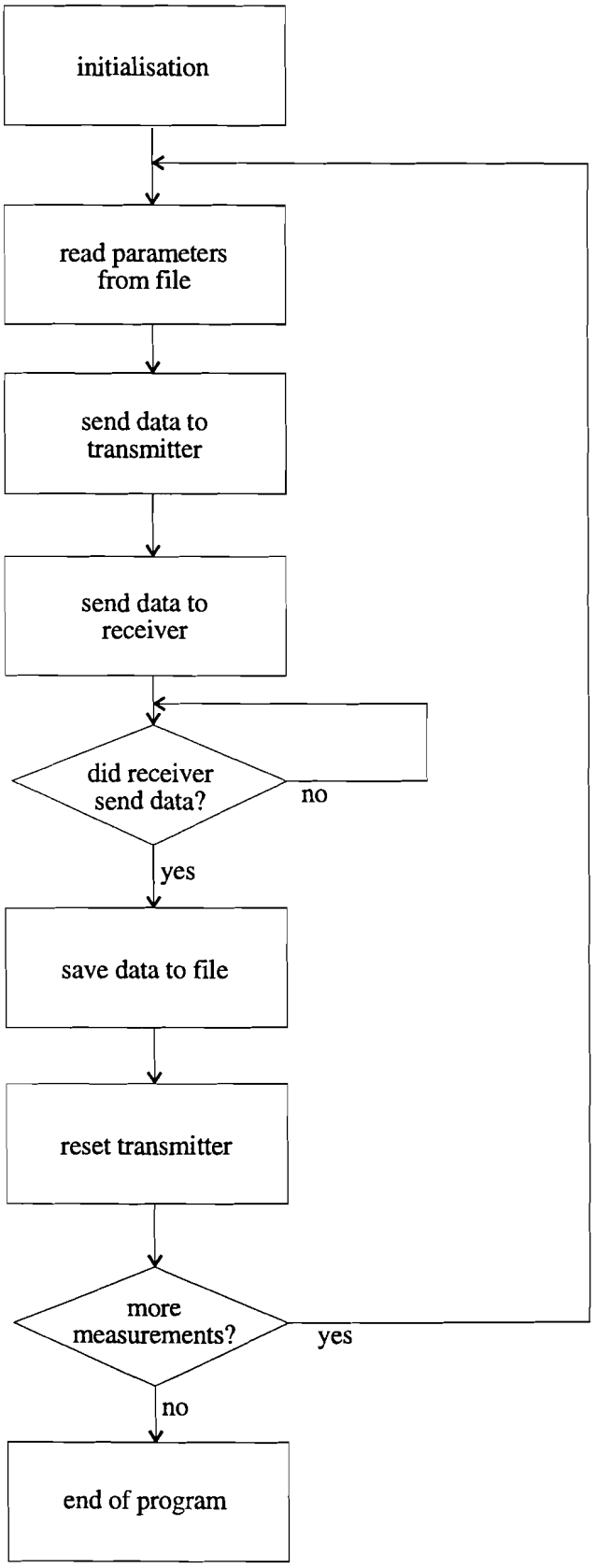


Figure 7.4: Flow chart of the personal computer program.

8 Measurements

8.1 Introduction

This chapter describes the acquisition-time measurements performed with the synchronous soft-decision single-dwell baseband acquisition system, already discussed in Section 4.3. Measurements are done for various signal-to-interference ratios. The measurement procedure is fully automatized and is controlled by a personal computer via the communications interface described in Chapter 6.

8.2 System overview

As described in Section 4.3, the acquisition-time measurements are performed with a synchronous soft-decision single-dwell baseband acquisition system. Measurements were done for a code length of $N=127$ and a chip rate of $f_c=16$ kcps. The dwell time is $\tau_d=NT_c=127T_c$. In future, measurements may be performed for code lengths of 255, 511, and 1023, and for higher chip rates. The aim of the measurements described in this chapter is to verify the correct operation of the system, to show how a single-dwell acquisition system behaves under various noise conditions, and to get an overview of the relationships between the system parameters.

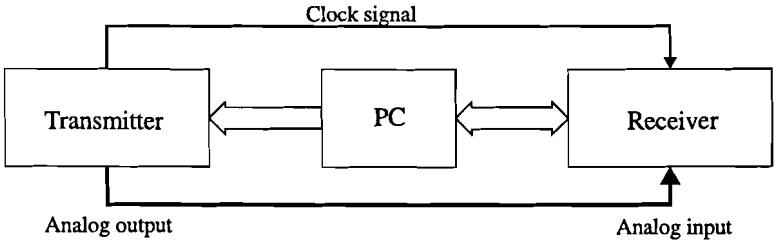


Figure 8.1: Overview of the measurement system.

Figure 8.1 shows an overview of the acquisition-time measurement system. The transmitter generates an analog signal consisting of the transmitter's own modulated data and random data modulated by a number of other users. Data are clocked out on a positive edge of the transmit-clock signal. This signal is also connected to the receiver to serve as a sample signal, thus providing a synchronous system. The analog transmitter output is sampled by the receiver on the negative edge of the clock signal. This enables the receiver to sample the transmitter output in the middle of a chip, which is the ideal case.

Each sample taken by the receiver is multiplied by a chip of the locally-generated PN sequence. After 127 samples are taken, a correlation is computed and compared to a threshold. If the correlation exceeds the threshold, the receiver gets into a verification mode. If, in verification mode, the correlation exceeds the threshold in the next two periods, acquisition is declared, else the receiver steps to the next cell and the procedure starts all over again. The receiver measures the time needed to acquire the correct code phase, i.e. the acquisition time. The measurement procedure is explained in the next section.

8.3 Measurement procedure

The measurement procedure is fully automatized and is controlled by the personal computer. The personal computer and the transceiver software were described in Chapter 6 concerning the communications interface. The main transceiver software was described in Chapter 7.

The measurement procedure can be divided into five steps:

1) the personal computer reads the system parameters from a parameter file on disk; the structure of the parameter file is shown in Appendix 5;
2) the system parameters are sent to the transmitter; the transmitter starts generating the required spread-spectrum signal, embedded in the signals of other users;
3) after a random time delay, the system parameters are also sent to the receiver; the receiver starts correlating the received signal with the locally-generated spreading code and does this until acquisition occurs; the random time delay provides a random code phase difference between the modulating sequence and the locally generated sequence;
4) if acquisition has occurred, the receiver despreads the transmitted data and sends the measured acquisition time, the number of false locks before acquisition, and the despreaded data to the personal computer;
5) the personal computer writes the acquisition data to file.

The data sent to the transmitter concern:

- the length of the feedback shift register with which the spreading code is generated; for the measurements presented in this chapter the shift-register length is 7; a 7-bit feedback shift register generates a maximal-length sequence of length N=127 (see Section 2.3);

- the phase of the spreading code relative to the spreading codes of the other users; this phase is a random value between 0 and 127 generated by the PC; it enables the transmitter to simulate a practical signal as well as possible;
- the number of other users of the system; this is a measure for the signal-to-interference ratio of the transmitter's output signal;
- the message that is modulated by the own spreading code; the data modulated by the spreading codes of the other users is random.

The data sent to the receiver concern:

- the length of the feedback shift register with which the local spreading code is generated;
- the threshold to which the correlation should be compared;
- the number of verifications; this value represents the number of successive periods in which the computed correlation must exceed the threshold before acquisition is declared.

8.4 System optimization

The detection probability P_D and the false-alarm probability P_{FA} have great influence on the acquisition-time performance. A variable system parameter that influences the detection and false-alarm probabilities is the threshold level η. Figure 8.2 illustrates the influence of η on P_D and P_{FA}. The left curve shows the probability density function (pdf) of the signal after correlation when the locally generated spreading code is not synchronized to the received signal. The signal after correlation is the signal at the output of the integrator in Figure 3.2. The double-crosshatched area represents P_{FA}. The right curve shows the pdf of the integrator output signal when the local spreading code is synchronized to the received signal. The crosshatched area represents P_D. If η is low, the detection probability is approximately unity, but also the false-alarm probability increases if the threshold decreases. This causes the detector to have many false locks and it becomes more probable that also the acquired code phase is false. If η is high, the false-alarm probability is very small and approaches zero, thus the number of false locks decreases. However, if the threshold setting is too high, the probability that the correlation exceeds the threshold for a number of successive periods becomes smaller. This causes the acquisition system to get out of the verification mode, even though the right code phase was detected.

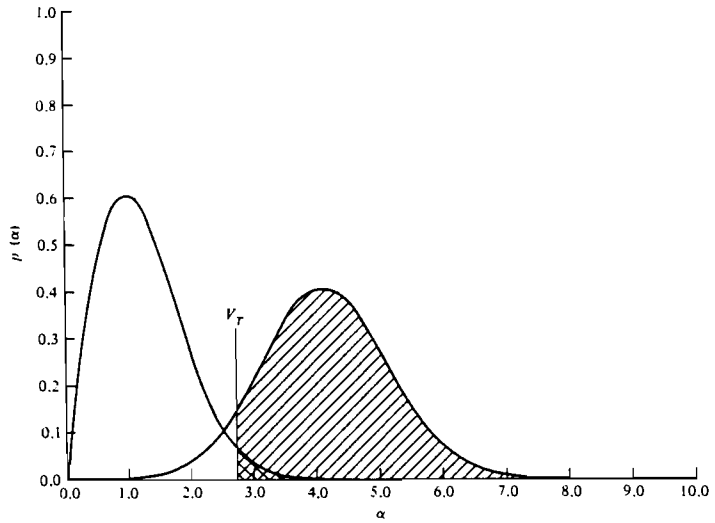


Figure 8.2: Threshold influence on P_D and P_{FA}.

Before acquisition measurements are performed, the threshold must be optimized to yield a minimum acquisition time for various signal-to-interference ratios. The optimization isdone by varying the threshold of the acquisition system and leaving all other system parameters and interference conditions equal. Figures 8.3 to 8.6 show some results of the optimization procedure. They show measurement results of the mean acquisition time, the percentage of false locks when acquisition occurred, and the mean number of verifications before acquisition is declared, respectively, as a function of the threshold relative to the maximum correlation peak.

Optimization measurements are performed with the number of other users as a parameter. Results are shown for 4, 8, and 12 other users, respectively. One hundred measurements are done for all different conditions. From the angularity of the curves for large numbers of users, it can be concluded that this number is too small for reliable results. However, the results are satisfactory for the system optimization.

Figure 8.3 shows a few typical characteristics. The mean acquisition time increases with increasing threshold and increasing number of users. If the threshold gets above a certain level, the mean acquisition time increases rapidly. The threshold setting for which this happens decreases with increasing number of users of the channel. According to Figure 8.3, a low threshold yields the smallest acquisition time. This, however, is not entirely true. Figure 8.4 confirms this. When the threshold becomes lower, the probability of exceeding this threshold increases rapidly. This is due to a high interference level, i.e., the signals of other users. For a spread-spectrum system, this implies that the acquisition system hands over to the tracking system, although the false code phase has been

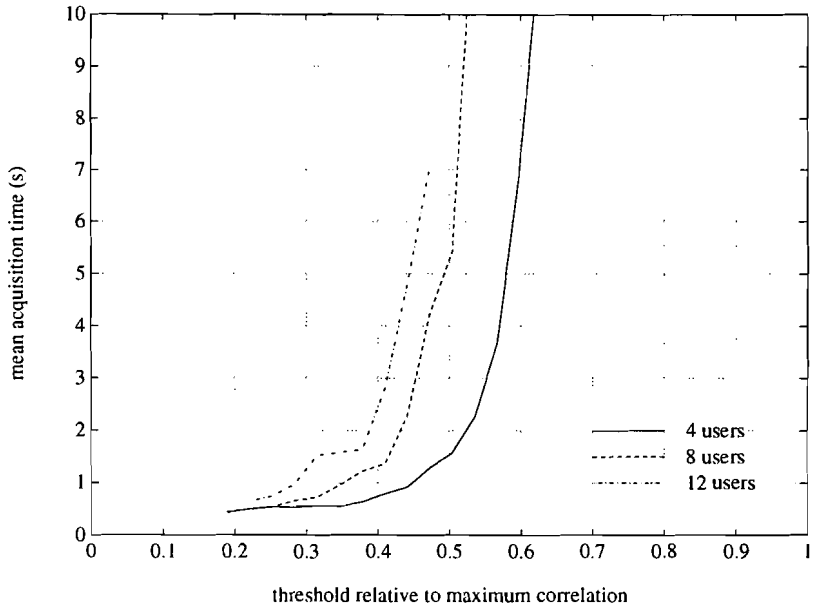


Figure 8.3: T_{ACQ} as a function of the threshold for different numbers of users.

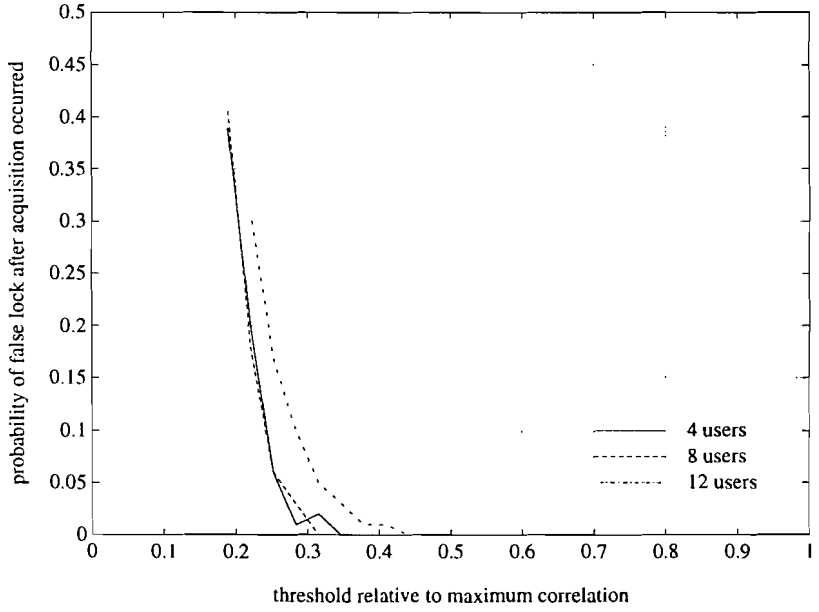


Figure 8.4: Probability of false locks after acquisition occurred as a function of the threshold, for different numbers of users.

acquired. The tracking system will soon get out of lock, and control is handed over to the acquisition system again. This whole procedure will cause the effective acquisition time to increase. The effective acquisition time is the time needed to acquire the correct code phase. This in contrast to the acquisition time as defined previously, in which the correlation exceeds the threshold for three successive periods.

The effective acquisition time is calculated using the results shown in Figures 8.3 and 8.4. It is calculated using the following formula:

$$T_{ACQ} = (1-P_{FL})T_{ACQ,\,measured} + \frac{P_{FL}}{1-P_{FL}} \cdot \left(T_{FL} + T_{ACQ,\,measured}\right), \tag{8.1}$$

where P_{FL} is the probability that the false code phase has been acquired, and T_{FL} is the time the tracking system needs to get out of lock after a false acquisition occurred. T_{FL} is assumed T_{FL}=0.3 s, just to illustrate the effect. As an example, it is computed for the case that the number of users is 4. Results are shown in Figure 8.5. From Figure 8.5, the optimum threshold is found for a relative threshold level of η=0.28.

Since no information about the tracking system is available, the optimization criterion as defined cannot be applied. To optimize the threshold of the acquisition system, another criterion must be used. The optimum threshold is defined as the minimum threshold for which the acquisition time is minimum and the probability of a false locks after acquisition occurred is below one per cent. Measurements performed with an acquisition system with this optimum threshold are presented in the next section.

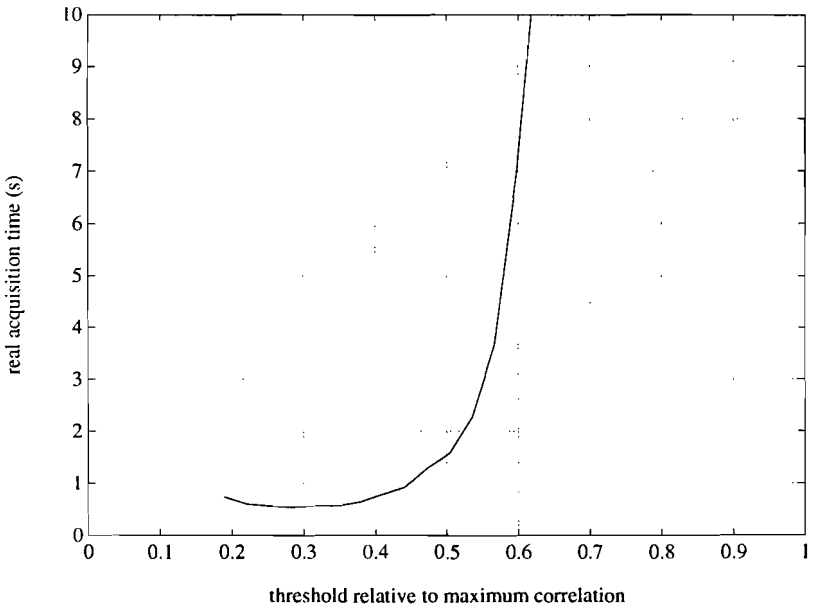


Figure 8.5: Effective acquisition time as a function of the threshold (4 other users).

Finally, Figure 8.6 shows the measurements of the number of verifications before acquisition is declared. These figures confirm the statements made in the beginning of this section. For low threshold settings, the number of false locks is large because the false alarm probability increases with decreasing threshold. Therefore the number of verifications before the correct code phase has been detected is also high. For high threshold settings, the number of verifications is large, because the detection probability

decreases with increasing threshold. This causes the acquisition system to get out of the verification mode, even though the correct code phase was detected. In between, a specific threshold yields a minimum number of verifications. It is not necessary that this minimum occurs at the same threshold level as the minimum acquisition time occurs. The figure also shows that the mean number of verifications increases as the number of other users increases. This is due to the fact that the detection probability decreases and the false-alarm probability increases with increasing number of other users.

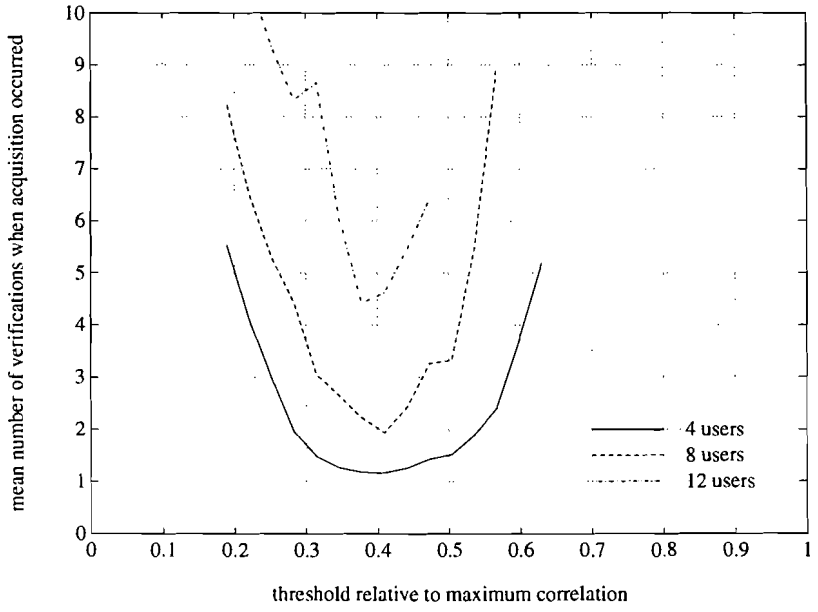


Figure 8.6: Mean number of verifications when acquisition occurs.

8.5 Results

After the detector threshold is optimized according to the procedure discussed in Section 8.4, the mean and variance of the acquisition time can be determined as a function of the number of other users, i.e. as a function of the signal-to-interference ratio at the input of the acquisition system. One thousand measurements are performed for all different signal-to-interference ratios to guarantee reliable results. These results are shown in Figures 8.7 and 8.8. They show that the mean and variance of the acquisition time increase when the number of users increases. This was expected, because the detection probability decreases and the false-alarm probability increases when the number of users increases. According to Equations (3.1) and (3.2), then the mean and variance of the acquisition time increase.

In practice, the acquisition-time requirement is specified by a certain probability that it is guaranteed that the system will acquire the correct code phase within a specific time. The

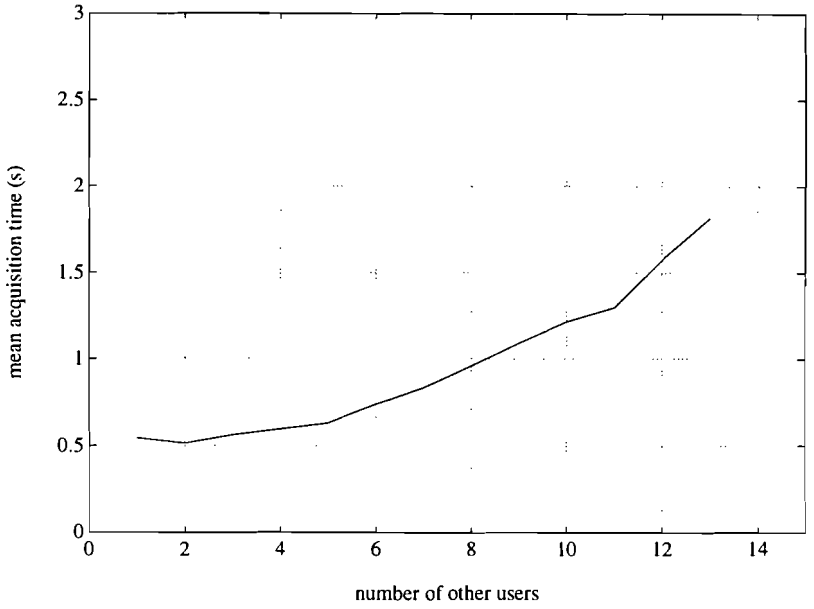


Figure 8.7: Mean acquisition time as a function of the number of other users. (N=127, f_c=16 kcps, η_d=127T_c, K=2).

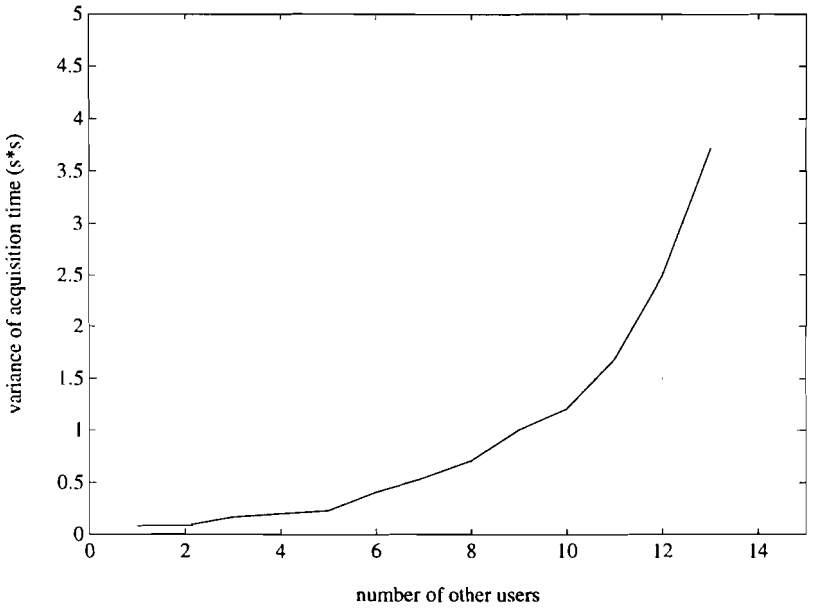


Figure 8.8: Variance of the acquisition time as a function of the number of other users.

probability that the acquisition time is below this specific value must exceed this required value. The mean acquisition time gives an indication whether this requirement can be satisfied or not. It must be less than this maximum time required. Besides the mean acquisition time, the variance of the acquisition time is an important system parameter. It indicates the spreading in the acquisition time. For the measurements performed, the variance of the acquisition time increases rapidly when the number of users exceeds

approximately 10. From a system point of view, 10 can be seen as the maximum number of users that may use the channel simultaneously. It is very likely that the system requirement is not satisfied if the number of simultaneous users exceeds 10.

In literature, e.g. in [5], it is shown that for a BPSK spread-spectrum system the interference of other users can be characterized by zero-mean Gaussian noise. If there are, e.g., 8 other users, the signal-to-interference ratio at the input of the acquisition system is 1/8 or -9 dB. This interference can be approximated by zero-mean Gaussian noise with equal power. Thus the signal-to-noise ratio at the input of the acquisition system is -9 dB when 8 users use the channel simultaneously. For a system with many users, the system noise can be neglected with regard to the interference.

8.6 Conclusions

Measurements were performed with the acquisition system implemented. The acquisition system uses the single-dwell serial search method to detect synchronization. The acquisition system was optimized to yield minimum acquisition time. This was done by optimising the threshold level of the comparator. Because the characteristics of the tracking system are not known, the practical optimization criterion could not be applied. Therefore, a simpler criterion was used to optimize the threshold.

With the optimized acquisition system, measurements of the mean and variance of the acquisition time were performed. Results show that the acquisition system behaves as expected. The performance decreases when the interference power increases. They also show that, for a code length of N=127, the maximum number of users of the channel is approximately 10. If the number of users is larger than 10, the acquisition system is not likely to satisfy the system requirement. The use of spreading codes with better cross-correlation properties, like Gold-codes [3], may increase the maximum number of simultaneous users.

Within the time constraints, it was not possible to derive a relation between the acquisition performance of a BPSK spread-spectrum system [5] and the performance of the baseband system implemented. Therefore, it is not possible to verify the measurements by comparing them to theory.

9 Conclusions and recommendations

9.1 Conclusions

For a satellite communications system using picoterminals, a baseband direct-sequence spread-spectrum transceiver based on a digital signal processor was developed and implemented. The spread-spectrum transmitter can generate a spreaded data signal embedded in interference of other channel users. This is used to test the acquisition behaviour of the receiver under various signal-to-interference conditions. On the receiver the single-dwell serial-search acquisition method was implemented. With the acquisition system, acquisition-time measurements were performed. The system was optimized to yield a minimum acquisition time. Results show that the acquisition-time performance decreases if the number of channel users increases. For a practical system, the number of channel users must stay below a certain value to guarantee correct operation.

It is assumed that the behaviour of the baseband acquisition system gives a good indication of how a practical direct-sequence BPSK system behaves. Within the given time, no relation was found between the performance analysis of a BPSK spread-spectrum system and the acquisition performance for the baseband implementation. Therefore, it is not possible to verify the measurements by comparing results and theory.

The transceiver designed for the implementation of the baseband spread-spectrum system can also be used as the baseband spread-spectrum modulator in a BPSK direct-sequence spread-spectrum system. The measurement system as implemented has many possible configurations, both in hardware as well as in software. This allows the user to test not only acquisition behaviour of direct-sequence spread-spectrum signals, also the tracking behaviour and bit-error rates can be studied. In fact, the transceiver can be used for many applications involving signal processing.

The electronic circuit of the transceiver was designed using the OrCAD design program. For the first time, the OrCAD program was also used to generate the data file for the design of the printed circuit board. Thus, after the design of the electronic circuit, the printed circuit board design was fully automatized.

While designing the printed circuit board, great care was taken for the positioning of components on the printed circuit board and routing of power-supply lines, ground lines,

and high-speed lines (data bus, address bus, etc.). This was done to reduce interference of components or connections to each other and to other printed circuit boards. It was demonstrated that these EMC measures are necessary for correct operation of an electronic circuit using high clock frequencies.

9.2 Recommendations

The system designed can be used for a variety of measurements to test the behaviour of spread-spectrum systems. For picoterminal communication systems, a large processing gain is needed, which implies the use of long spreading codes. To study the behaviour of a system using these large spreading codes, measurements can be performed with respect to acquisition and tracking behaviour of such a system. Also the use of other codes than maximal-length sequences can be considered. In addition, the performance of other acquisition techniques can be tested and compared to the performance of the single-dwell acquisition system.

The use of the OrCAD design program for electronic circuits to generate the data file for the printed circuit board design is not recommended, unless the user is familiar with this program and has some experience in designing electronic circuits. Otherwise, failures can be made which cause incorrigible errors in the printed circuit board design.

In future, the EMC considerations as mentioned in Section 5.13 and applied to the printed circuit board design must be applied for electronic circuits using high clock frequencies to assure correct operation and minimum interfering radiation.

10 References

[1] Sage, G.F., SERIAL SYNCHRONISATION OF PSEUDONOISE SYSTEMS, IEEE Transactions on Communication Technology, vol.12 (December 1964), p.123-127.

[2] Ziemer, R.E. and R.L. Peterson, DIGITAL COMMUNICATIONS AND SPREAD SPECTRUM SYSTEMS, Macmillan Publishing Company, 1985.

[3] Holmes, J.K., COHERENT SPREAD SPECTRUM SYSTEMS, John Wiley & Sons, 1982.

[4] Holmes, J.K. and C.C. Chen, ACQUISITION TIME PERFORMANCE OF PN SPREAD SPECTRUM SYSTEMS, IEEE Transactions on Communications, vol.25 (1977), no.8, p.778-783.

[5] Simon, M.K, J.K. Omura, R.A. Scholtz, and B.K. Levitt, SPREAD SPECTRUM COMMUNICATIONS, Volumes I, II, and III, Computer Science Press, 1985.

[6] SECOND-GENERATION TMS320 USER'S GUIDE, Texas Instruments, 1987.

[7] TMS 320 SECOND-GENERATION DIGITAL SIGNAL PROCESSORS - ADVANCE INFORMATION, Texas Instruments, May 1987 - revised May 1989.

[8] DIGITAL SIGNAL PROCESSING APPLICATIONS WITH THE TMS320 FAMILY: THEORY, ALGORITHMS AND APPLICATIONS - Volume 2, Texas Instruments, 1990.

[9] Martinino, F., A. Pugnaloni, A. Saitto, and M. Tripodi, ERROR PROBABILITY EVALUATION FOR A DIGITAL SPREAD SPECTRUM MODEM USED IN SATELLITE COMMUNICATIONS, Alta Frequenza, vol.57 (1988), no.1, p.21-28.

[10] Ginsberg, G.L., PRINTED CIRCUITS DESIGN: FEATURING COMPUTER-AIDED TECHNOLOGIES, McGraw-Hill Inc., 1991.

[11] Strik, M.T.J., DIGITAL BEACON RECEIVER FOR PROPAGATION EXPERIMENTS, PART II DECIMATOR/DETECTOR FOR PHASE AND AMPLITUDE MEASUREMENT, Graduation Report EUT, Telecommunications Division, 1992.

[12] Seyer, M.D., COMPLETE GUIDE TO RS232 AND PARALLEL CONNECTIONS: A STEP-BY-STEP APPROACH TO CONNECTING COMPUTERS, PRINTERS, TERMINALS, AND MODEMS, Prentice-Hall, 1988.

[13] Brown, R. and J. Kyle, PC INTERRUPTS: A PROGRAMMER'S REFERENCE TO BIOS, DOS, AND THIRD-PARTY CALLS, Addison-Wesley, 1991.

[14] Hyman, M.I., ADVANCED DOS: MEMORY-RESIDENT UTILITIES, INTERRUPTS, AND DISK MANAGEMENT WITH MS- AND PC-DOS, MIS (Portland, Oregon), 1989.

[15] TURBO PASCAL VERSION 7.0 USER'S GUIDE, LANGUAGE GUIDE, AND PROGRAMMER'S REFERENCE, Borland International, 1992.

Appendix 1: The electronic circuit

This appendix gives an overview of the hardware designed. Figures A.1.1, A.1.2, and A.1.3 show the transceiver circuit designed. Figure A.1.2 shows the digital signal processor, reset circuit, oscillator, wait-state generator, EPROMs, and RAMs. In Figure A.1.3, the communications interface with DIP-switches, the I/O selection circuit, the buffered DAC, and the ADC are shown. In Figure A.1.1, these two sheets are connected in an overlay sheet and the input and output connectors are defined.

Table A.1.1 gives a list of the components used in the transceiver. To give an overview of all possible configurations, the settings of all jumpers are listed in Table A.1.2. All pins of the 64-pins connector are listed in Table A.1.3. Also the purpose of the input or output pins is indicated. A listing of the DIP-switch settings can be found in Chapter 7.

The hardware layout is illustrated in Figures A.1.4, A.1.5, and A.1.6. These figures show the placement of the components on the printed circuit board (PCB), and the component and solder sides of the designed PCB, respectively.

All sheets are designed using the OrCAD design program for electronic circuits. To design a printed circuit board for the transceiver circuit, OrCAD can generate a so called netlist, in which all parts of the circuit and all connections between parts are listed in a specific format. The actual design of the printed circuit board is done at the Central Technical Service (CTD) of the Eindhoven University of Technology. They can read the 'EDIF' and the 'SciCards' netlist formats.

If one does not have any experience in designing circuits with OrCAD, it is better to deliver the printouts of the sheet(s) to the CTD. Because a connection on the sheet not always means a connection in the netlist, it is better to not deliver a netlist to the CTD. However, if one has good experience in working with OrCAD, it can be time-saving to deliver the netlist.

The printed circuit board is designed in close cooperation with the CTD. They know the different possibilities to design a PCB, whereas the designer of the hardware knows the functions of the different components and where to place them. Special wishes should be mentioned before the CTD starts designing the PCB.

Figure A.1.3: The communicatio
buffered DAC, and

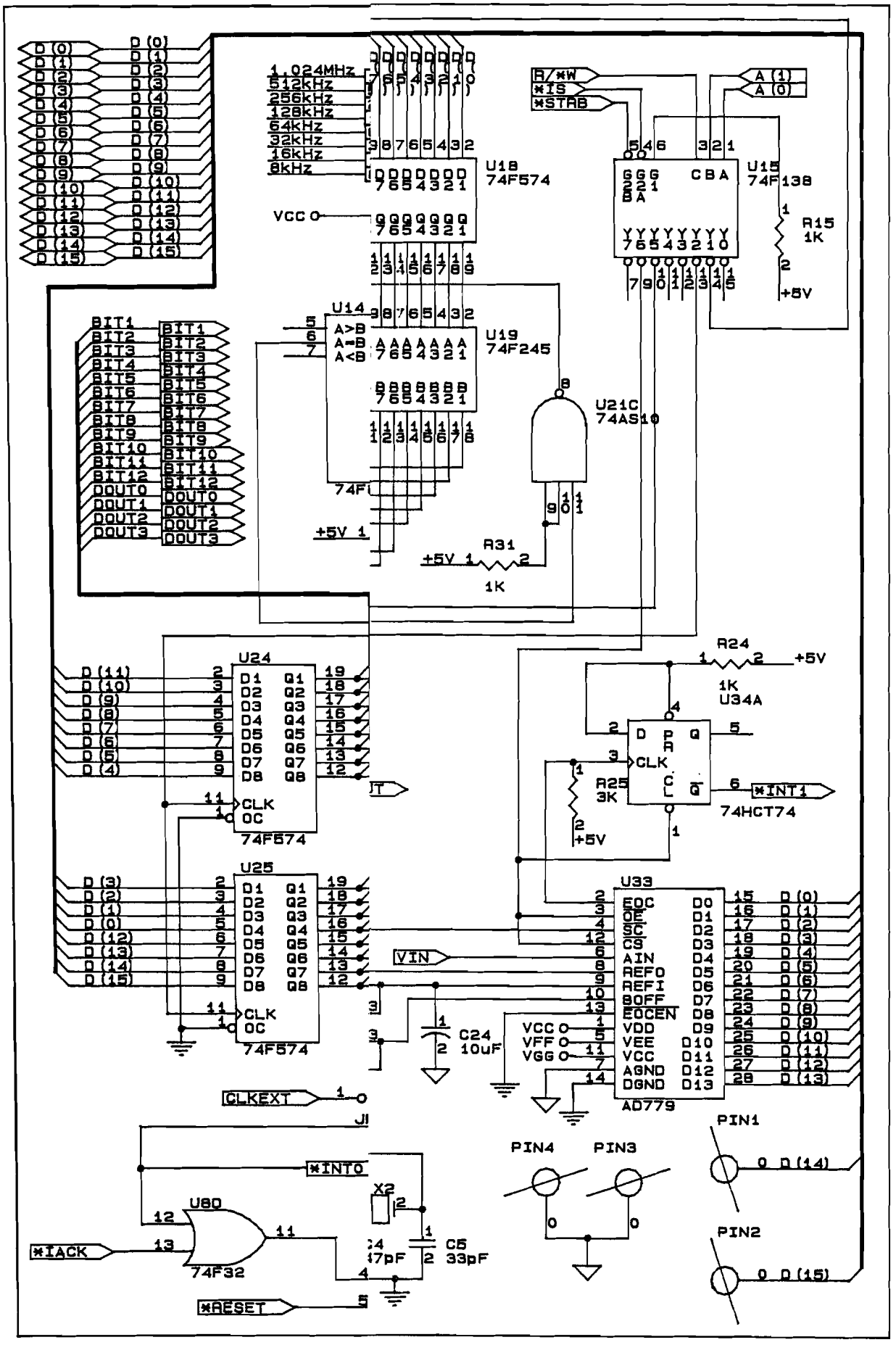


Figure A.1.2: The digital signa
generator, EPRO

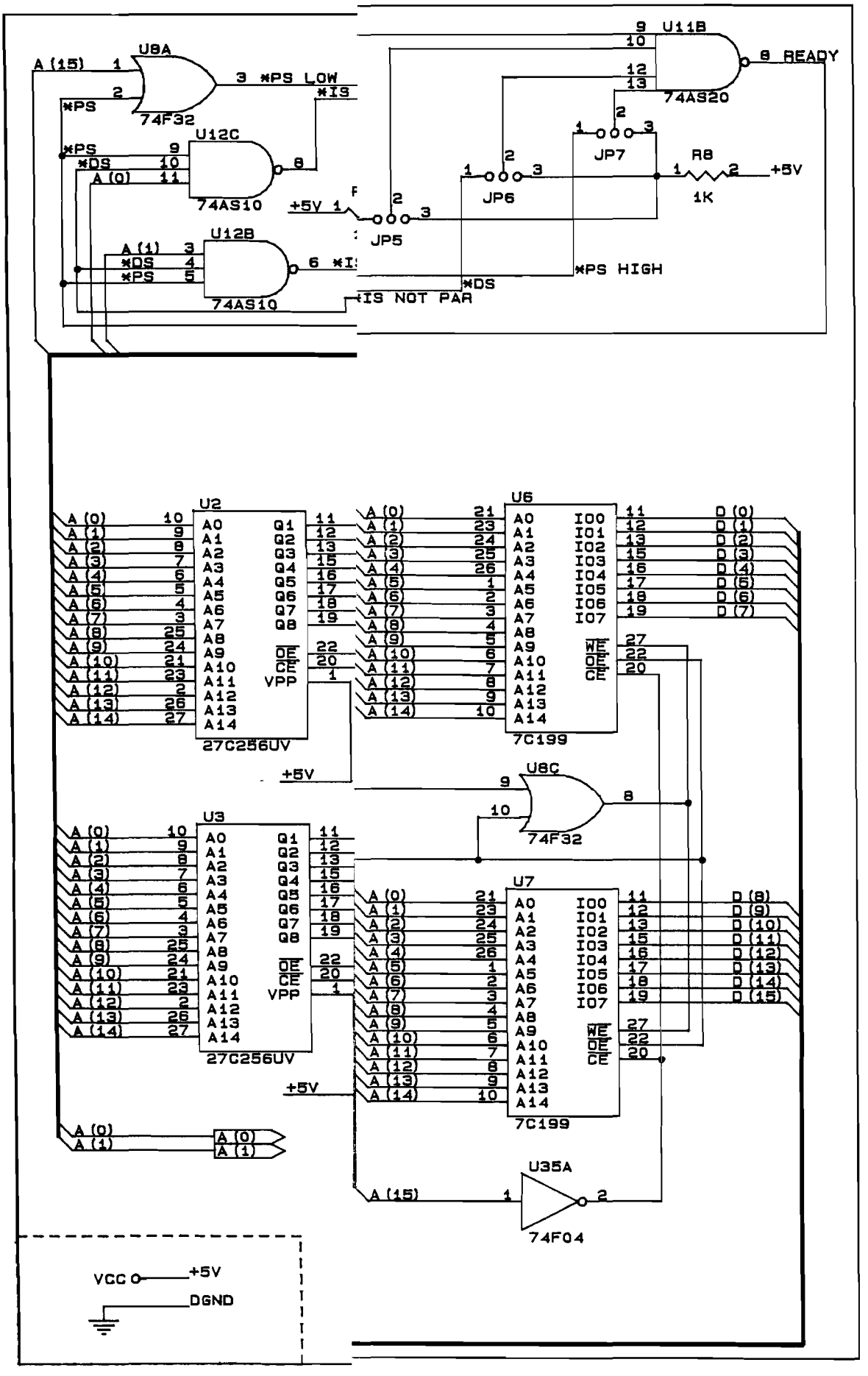


Figure A.1.1: The overlay sheet wi

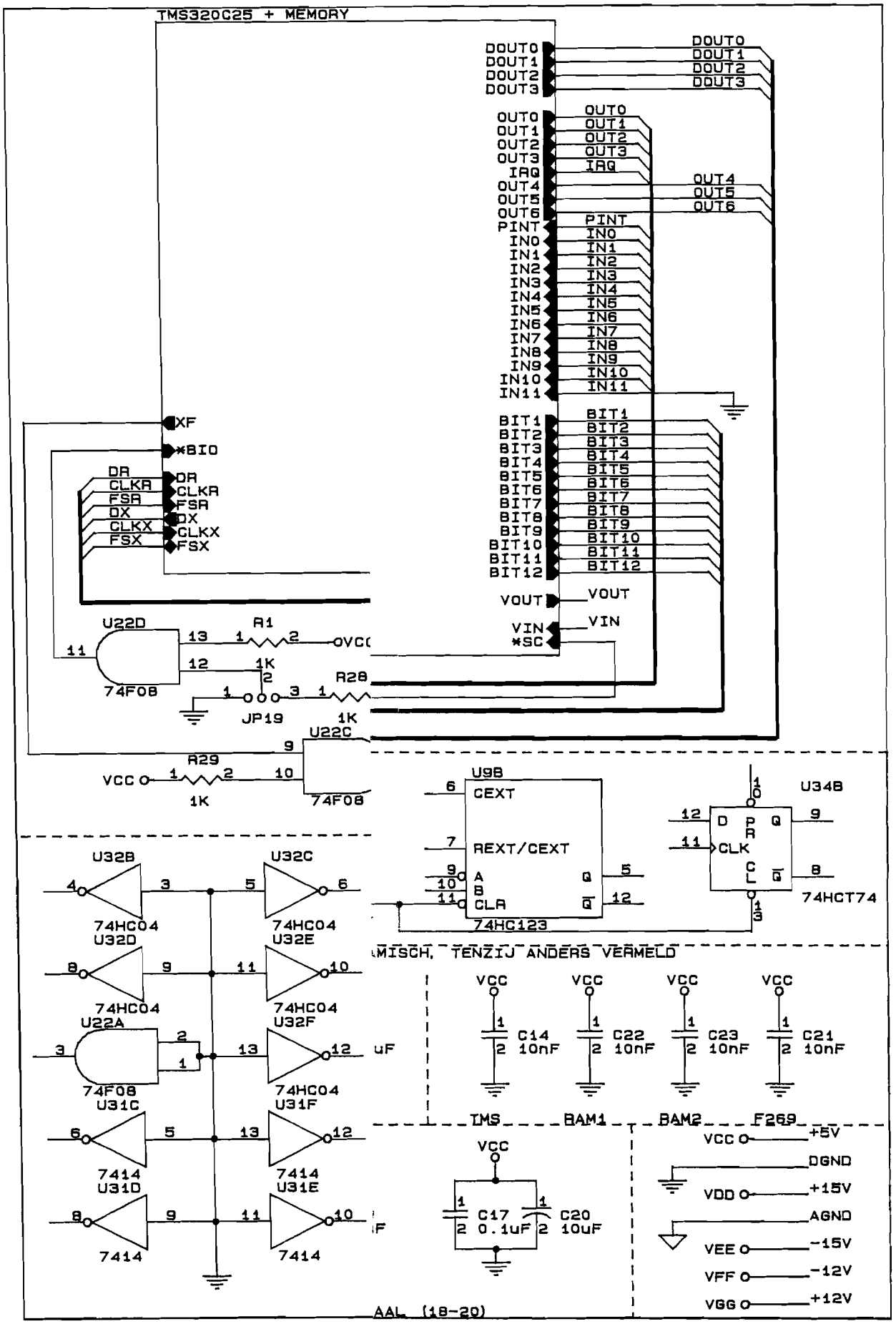


Table A.1.1: List of the components used.

Item	Quantity	Reference	Part
1	1	C1	Electrolytic 0.47uF
2	2	C2,C3	Ceramic 10pF
3	1	C4	Ceramic 47pF
4	1	C5	Ceramic 33pF
5	1	C6	Ceramic 7pF
6	9	C7,C8,C10,C11,C12,C13, C15,C16,C17	Ceramic 0.1uF
7	5	C9,C14,C21,C22,C23	Ceramic 10nF
8	1	C24	Tantalum 10uF
9	3	C18,C19,C20	Tantalum 10uF
10	1	C25	Ceramic 100pF
11	1	J1	Connector 2x32 pins
12	2	J2,J3	BNC plug
13	19	JP1,JP2,JP3,JP4,JP5,JP6, JP7,JP8,JP9,JP10,JP11, JP12,JP13,JP14,JP15,JP16, JP17,JP18,JP19	Jumper
14	4	PIN1,PIN2,PIN3,PIN4	Extended IC-socket
15	23	R1,R2,R3,R4,R6,R7,R8,R9, R10,R11,R12,R13,R14,R15, R16,R17,R18,R24,R26,R28, R29,R30,R31	1k
16	1	R5	100k
17	1	R19	8k2
18	1	R20	1OM
19	1	R21	200
20	2	R22,R23	100
21	1	R25	3k
22	1	R27	10k
23	1	RP1	R-Pack
24	1	S1	Reset Button
25	1	S2	DIP-switches
26	1	U1	TMS320C25-50
27	2	U2,U3	27C256UV
28	5	U4,U5,U16,U17,U19	74F245
29	2	U6,U7	CY7C199
30	1	U8	74F32
31	1	U9	74HC123
32	1	U10	74F114
33	1	U11	74AS20
34	2	U12,U21	74AS10
35	1	U13	Oscillator 50 MHz
36	1	U14	74F85
37	1	U15	74F138
38	5	U18,U24,U25,U26,U27	74F574
39	2	U20,U34	74F74
40	1	U22	74LS08
41	1	U23	74F269
42	1	U28	74F151
43	1	U29	AD568JQ
44	1	U30	AD840JN
45	1	U31	74LS14
46	1	U32	74HCU04
47	1	U33	AD779JN
48	1	U35	74F04
49	2	X1,X2	Crystal 20 MHz

Table A.1.2: Jumper settings for the transceiver.

JP1 to JP7 - jumper settings for the wait-state generator:

Memory or peripheral	Number of wait-states	Jumper settings JP1 to JP7						
		JP1	JP2	JP3	JP4	JP5	JP6	JP7
EPROM	2	-	-	-	-	-	-	-
Program RAM	0	-	-	-	23	-	-	12
	1	-	-	-	12	-	-	23
Data RAM	0	-	-	23	-	-	12	-
	1	-	-	12	-	-	23	-
I/O Par.prt.	2	-	-	-	-	-	-	-
I/O not Par.prt.	0	23	23	-	-	12	-	-
	1	23	12	-	-	23	-	-
	2	12	23	-	-	23	-	-

JP8 - jumper setting for selection of the processor clock:

Crystal oscillator	12
Internal oscillator	23

JP9, JP10 - jumpers for selection of the communications interface device:

Number of devices connected to parallel port of PC	Jumper settings JP9 and JP10	
	JP9	JP10
0 - 2	23	23
3 - 4	12	23
4 - 8	12	12

If pins 2 and 3 of JP9 and JP10 are connected, the device number is set by DIP-switches 7 and 8. If pins 1 and 2 of JP9 and pins 2 and 3 of JP10 are connected, the device number can be set by DIP-switches 6 to 8. If pins 1 and 2 of both JP9 and JP10 are connected, DIP-switches 5 to 8 set the device number.

JP11 - jumper for selection of the transmit clock:

Internal transmit clock	23
External transmit clock	12

JP12 to JP14 - jumpers for selection of the internal transmit clock frequency:

Selection by DIP-switches 1 to 3	12
Selection by software	23

JP15, JP16, and JP18 - jumpers to select if the *SC signal is to be transformed:

	JP15	JP16	JP18
The *SC signal is not transformed	23	12	12
The *SC signal is transformed	23	23	23

If the sample frequency is 128 kHz, the *SC input of the ADC can be grounded. This means that pins 1 and 2 of jumper JP15 are connected. Jumpers JP16 and JP18 do not care in this case.

JP17 - jumper to select the *SC signal:

The *SC signal is provided at the 64-pins connector	12
The *SC signal is provided by the internal transmit clock	23

JP19 - jumper to set the *BIO input of the processor:

The *BIO input is low	12
The *BIO input is high	23

Table A.1.3: Connector pin names.

Pin	Pin name	Function	Pin number	Pin name	Function
A1	DOUT0	Programm.	C1	VDD	+15 V
A2	DOUT1	Programm.	C2	VEE	-15 V
A3	DOUT2	Programm.	C3	VFF	+12 V
A4	DOUT3	Programm.	C4	VGG	-12 V
A5	OUT4	Programm.	C5	AGND	An.Grnd
A6	OUT5	Programm.	C6	VCC	+5 V
A7	OUT6	Programm.	C7	GND	Dig.Grnd
A8	XF	XF output	C8	NC	
A9	8 kHz	Int. Clock	C9	NC	
A10	16 kHz	Int. Clock	C10	NC	
A11	32 kHz	Int. Clock	C11	NC	
A12	64 kHz	Int. Clock	C12	NC	
A13	128 kHz	Int. Clock	C13	NC	
A14	256 kHz	Int. Clock	C14	NC	
A15	512 kHz	Int. Clock	C15	NC	
A16	1.024 MHz	Int. Clock	C16	NC	
A17	CLKEXT	Ext. Clock	C17	NC	
A18	*SC	Start Conv	C18	NC	
A19	FSX	Ser.Port	C19	NC	
A20	CLKX	Ser.Port	C20	PINT	Comm.Int
A21	DX	Ser.Port	C21	IN0	Comm.Int
A22	FSR	Ser.Port	C22	IN1	Comm.Int
A23	CLKR	Ser.Port	C23	IN2	Comm.Int
A24	DR	Ser.Port	C24	IN3	Comm.Int
A25	IN8	Comm.Int	C25	IN4	Comm.Int
A26	OUT0	Comm.Int	C26	IN5	Comm.Int
A27	IN9	Comm.Int	C27	IN6	Comm.Int
A28	IN10	Comm.Int	C28	IN7	Comm.Int
A29	VCC	+5 V	C29	IRQ	Comm.Int
A30	VCC	+5 V	C30	OUT3	Comm.Int
A31	GND	Dig.Grnd	C31	OUT2	Comm.Int
A32	GND	Dig.Grnd	C32	OUT1	Comm.Int

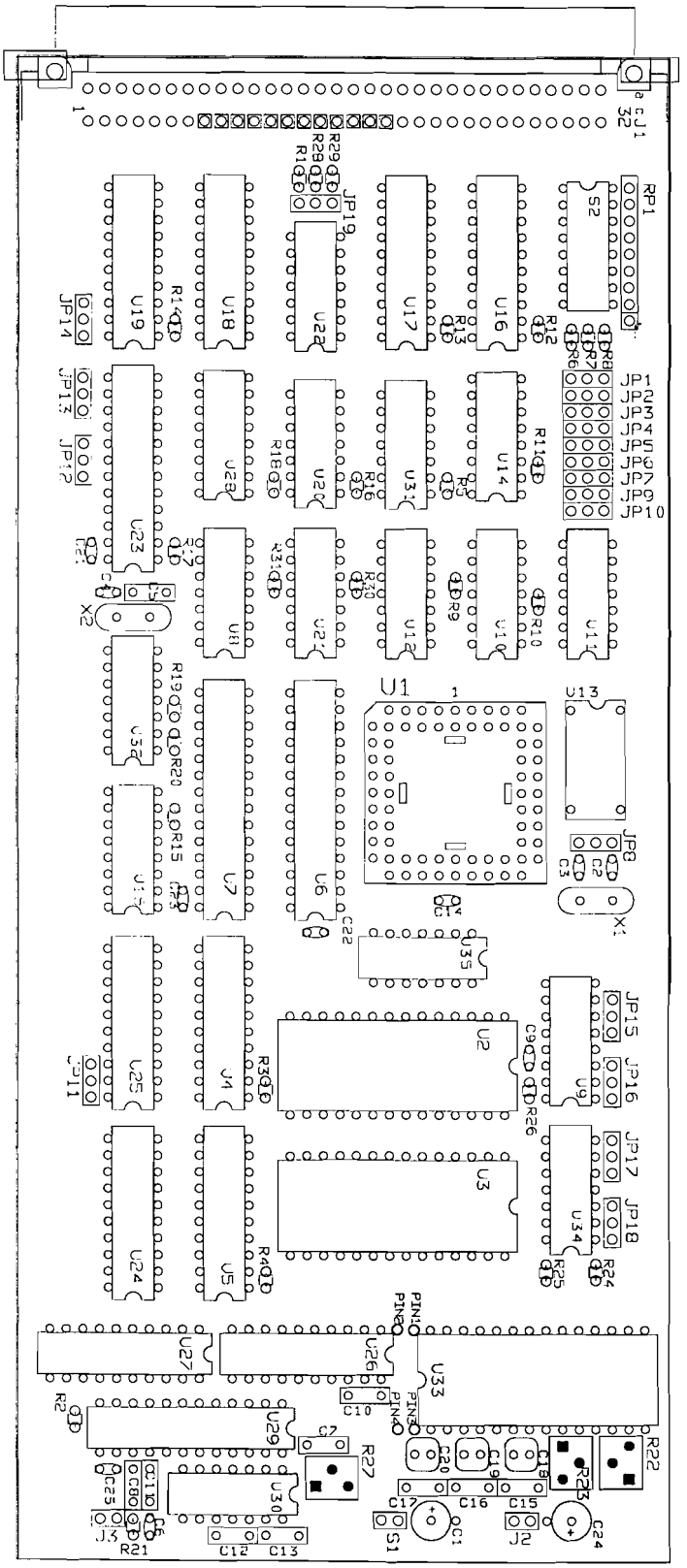


Figure A.1.4: Placement of components.

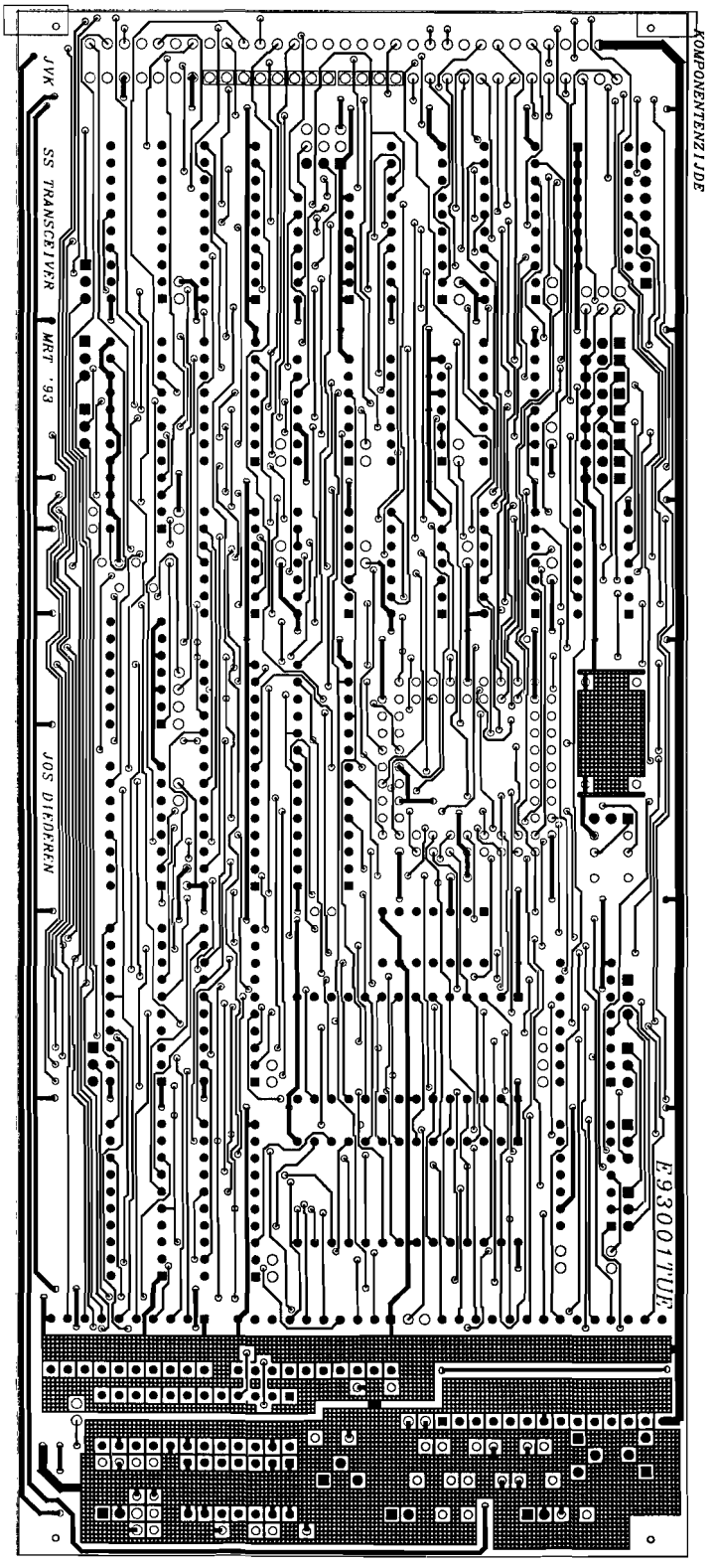


Figure A.1.5: The component-layer runs.

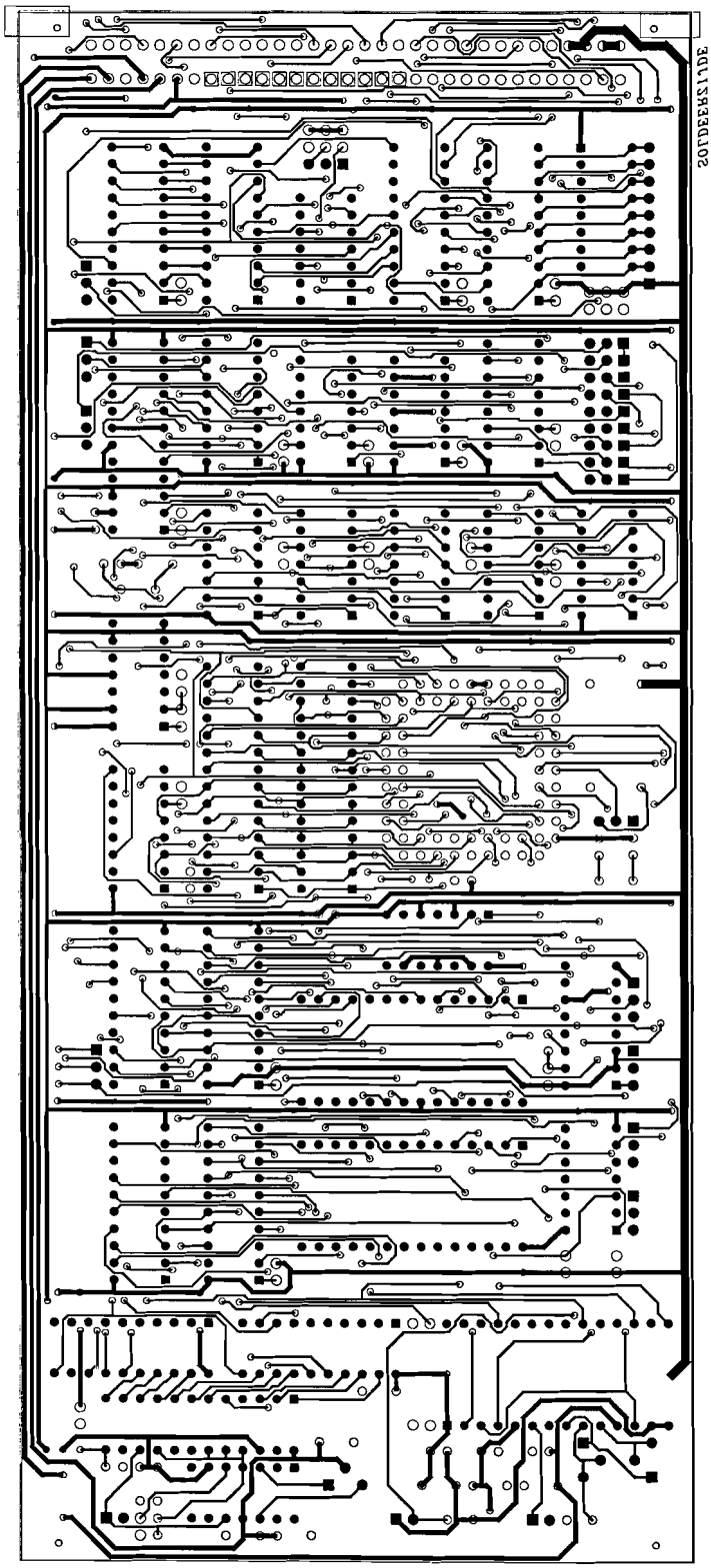


Figure A.1.6: The solder-layer runs.

Appendix 2: Data sheets

This appendix comprises the data sheets of the most important components used. These are:

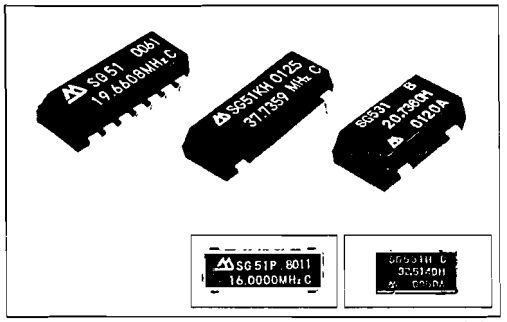
- Epson SG51KT/50.000 MHz crystal oscillator
- Cypress 7C199-25DC static RAM
- 27C256-UV/100ns EPROM
- Analog Devices AD568 digital-to-analog converter
- Analog Devices AD840 operational amplifier
- Analog Devices AD779 analog-to-digital converter

FULL SIZE DIP HIGH FREQUENCY CRYSTAL OSCILLATOR

SG-51 series

HALF SIZE DIP HIGH FREQUENCY CRYSTAL OSCILLATOR

SG-531 series



Actual size

■Specifications (characteristics)

Item		Symbol	SG-51/51K/51P/51E, SG-531/531P	SG-51T/51KT/51PT, SG-531T/531PT	SG-51KT/51PT, SG-531T/531PT	Remarks
			Specifications			
Output frequency range		fo	1.0250MHz to 26.0000MHz	26.0001MHz to 36.0000MHz	36.0001MHz to 50.3500MHz	
Power source voltage	MAX. supply voltage	V_{DD}-GND	−0.3V to +7.0V	−0.3V to +7.0V	−0.3V to +7.0V	
	Operating voltage	V_{DD}	5.0V±0.5V	5.0V±0.5V	5.0V±0.5V	
Temperature range	Storage temp.	T_{STG}	−55°C to +125°C	−55°C to +100°C	−55°C to +100°C	
	Operating temp.	T_{OPR}	−10°C to +70°C	−10°C to +70°C	−10°C to +70°C	
Soldering condition (lead part)		T_{SOL}	Under 260°C within 10 sec.	Under 260°C within 10 sec.	Under 260°C within 10 sec.	Package less than 150°C
Frequency stability		Δf/fo	B : ± 50ppm, C : ±100ppm	B : ± 50ppm, C : ±100ppm	B : ± 50ppm, C : ±100ppm	−10°C to +70°C
Current consumption		Iop	25mA MAX.	35mA MAX.	50mA MAX.	No load condition
Duty		Tw/T	40% to 60% * (45% to 55% *1)	40% to 60% ** (45% to 55% *1)	40% to 60% **	* : 1.4V or 1/2 V_{DD} level ** : 1.4V level
Output voltage		V_{OH}	V_{DD}−0.4V MIN.	V_{DD}−0.4V MIN.	2.4V MIN.	I_{OH}=−400μA
		V_{OL}	0.4V MAX. *	0.4V MAX. *	0.4V MAX. **	*: I_{OL}=16mA, **: I_{OL}=8mA
Output load condition (fan out)	TTL	N	10 TTL MAX.	10 TTL (30pF) MAX.	5 TTL (15pF) MAX.	
	C-MOS	CL	50pF MAX.			
Output enable/standby input voltage		V_{IH}	2.0V MIN.	2.0V MIN.	2.0V MIN.	
		V_{IL}	0.8V MAX.	0.8V MAX.	0.8V MAX.	
Output disable current		I_{OE}	12mA MAX.	20mA MAX.	25mA MAX.	OE=GND
Standby current		I_{ST}	310μA MAX.			$\overline{ST}$=GND
Output rise time		t_{TLH}	8nsec. MAX.	10nsec. MAX.	6nsec. MAX.	Refer to output waveform chart (page 9)
Output fall time		t_{THL}	8nsec. MAX.	8nsec. MAX.	6nsec. MAX.	
Oscillation start time		t_{OSC}	4msec. MAX.	10msec. MAX.	10msec. MAX.	More than for 1ms until V_{DD}=0V→4.5V Time at 4.5V to be 0sec.
Aging		fa	±5ppm/year MAX.	±5ppm/year MAX.	±5ppm/year MAX.	Ta=25°C, V_{DD}=5V, first year
Shock resistance		S. R.	±20ppm MAX.	±20ppm MAX.	±20ppm MAX.	Drop test of 3 times on a hard board from 75cm height or excitation test with 3000G×0.3ms×1/2 sine wave in 3 directions.

Note: · Unless otherwise stated, characteristics (specifications) shown in the above table are based on the rated operating temperature and voltage condition.
· External by-pass capacitor is recommended.

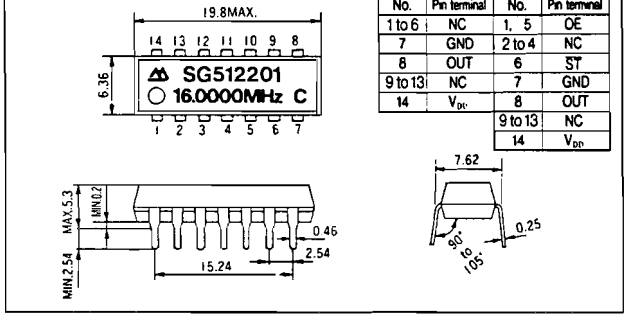
■Frequency correspondence table

Model \ Frequency	1MHz	26MHz	50MHz	66.67MHz
SG-51/51K/51E,SG-531/531P	▼	▼	▼	▼
SG-51T/51KT/51ET,SG-531T/531PT	▼	▼	▼	▼ (SG-51T: Up to 36MHz)
SG-51PH/51YH,SG-531PH/531YH	▼	▼	▼	▼

■External Dimensions

(Unit : mm)

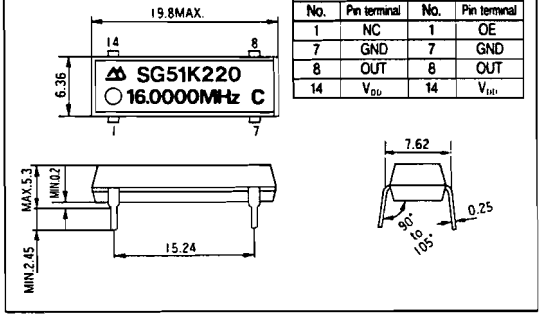
●SG-51/51T/51E



SG-51/51T		SG-51E	
No.	Pin terminal	No.	Pin terminal
1 to 6	NC	1, 5	OE
7	GND	2 to 4	NC
8	OUT	6	ST
9 to 13	NC	7	GND
14	V_{DD}	8	OUT
		9 to 13	NC
		14	V_{DD}

(Unit : mm)

●SG-51K/51KT/51YH/51P/51PT/51PH



SG-51K/51KT/51YH		SG-51P/51PT/51PH	
No.	Pin terminal	No.	Pin terminal
1	NC	1	OE
7	GND	7	GND
8	OUT	8	OUT
14	V_{DD}	14	V_{DD}

■Features

SG-51 series

- Pin compatible with full size metal can
- Packaged in plastic 14 pin DIP
- Auto insertable
- Provided with output enable and standby functions

SG-531 series

- Pin compatible with half size metal can
- Provided with output enable function

Common

- Cylindrical type AT cut quartz crystal built-in, thus assuring high reliability
- Possible with 386 CPU
- Use of C-MOS IC enables reduction of current consumption

Item		Symbol	SG-51PH/51YH,SG-531PH/531YH	
			Specifications	Remarks
Output frequency range		fo	26.0001MHz to 66.6667MHz	
Power source voltage	MAX. supply voltage	V_{DD}-GND	−0.3V to +7.0V	
	Operating voltage	V_{DD}	5.0V ±0.5V ※2	
Temperature range	Storage temperature	V_{STG}	−55°C to +100°C	
	Operating temperature	T_{OPR}	−10°C to +70°C	
Soldering condition (lead part)		T_{SOL}	Under 260°C within 10sec	Package less than 150°C
Frequency stability		Δf/fo	(B: ±50ppm)C ±100ppm	−10°C to −70°C. B type is possible up to 55MHz, please consult us.
Current consumption		Iop	35mA MAX.	No load condition Up to 45MHz : 21mA MAX.
Duty		Tw/T	40% to 60%	1/2 V_{DD} level
Output voltage		V_{OH}	V_{DD} −0.4V MIN.	I_{OH} = −4mA
		V_{OL}	0.4V MAX.	I_{OL} = 4mA
Output load condition(Fan out)	TTL	N		
	C-MOS	CL	50pF MAX.	
Output enable/standby input voltage		V_{IH}	2.0V MIN.	
		V_{IL}	0.8V MAX.	
Output disable current		I_{OE}	20mA MAX.	OE = GND. Up to 45MHz : 15mA MAX.
Standby current		I_{ST}		
Output rise time		t_{TLH}	7nsec.MAX. ※2	Over 45MHz : 5ns. MAX. Refer to output waveform chart (page 9)
Output fall time		t_{THL}	7nsec.MAX. ※2	
Oscillation start time		t_{OSC}	10msec.MAX.	More than for 1ms until V_{DD}=0V→4.5V Time at 4.5V to be 0sec.
Aging		fa	±5ppm/year MAX.	Ta=25°C V_{DD}=5V, first year
Shock resistance		S.R.	±20ppm MAX.	Drop test of 3 times on a hard board from 75cm height or excitation test with 3000G×0.3ms×1/2 sine wave in 3 directions in 3directions

※1 It is possible depending on condition, reference data (page 24).

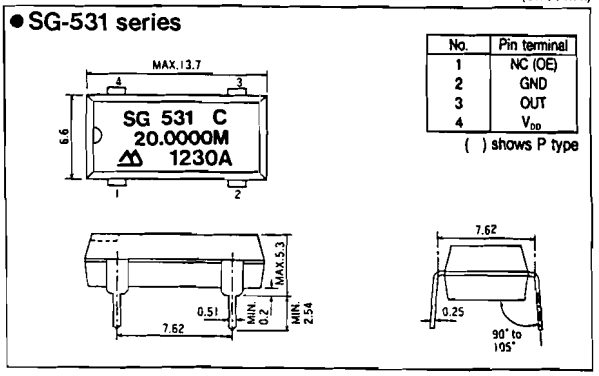
※2 AC characteristics of 386 CPU. (V_{DD}=5V ±0.25V, Load : CL≦50pF, Ta=−10 to +70°C, Refer to output waveform chart of 386 CPU).

Output frequency		26.001MHz to 36.000MHz		40.000MHz		45.000MHz to 50.000MHz		50.001MHz to 66.667MHz		Unit	Remarks
Item	Symbol	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	MIN.	MAX.		
CLK high time	t2a	9		8		7		6.25		ns	2V level
CLK high time	t2b	5		5		4		4.5		ns	Under 45MHz : V_{DD} −0.8V level Over 45MHz : 3.7V level
CLK low time	t3a	9		8		7		6.25		ns	2V level
CLK low time	t3b	7		6		5		4.5		ns	2v level
CLK fall time	t4		8		8		7		4	ns	Under 45MHz : V_{DD} −0.8V to 0.8V Over 45MHz : 3.7V to 0.8V
CLK rise time	t5		9		9		7		4	ns	Under 45MHz : V_{DD} 0.8V to V_{DD} −0.8V Over 45MHz : 0.8V to 3.7V

■External Dimensions

(Unit : mm)

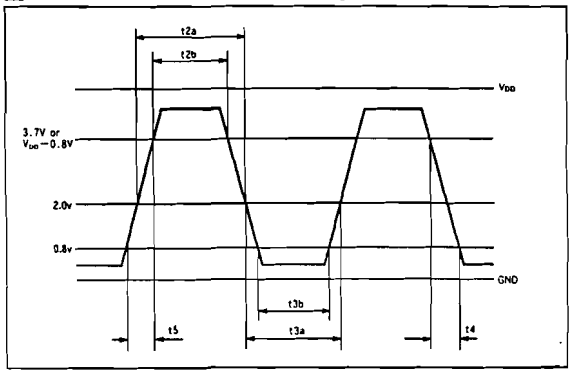
● SG-531 series



No.	Pin terminal
1	NC (OE)
2	GND
3	OUT
4	V_{DD}

( ) shows P type

■Waveform Chart of 386 CPU



CY7C198
CY7C199

CYPRESS SEMICONDUCTOR

32,768 x 8 Static R/W RAM

Features

- Automatic power-down when deselected
- CMOS for optimum speed/power
- High speed
  — 25 ns
- Low active power
  — 880 mW
- Low standby power
  — 220 mW
- TTL-compatible inputs and outputs
- Capable of withstanding greater than 2001V electrostatic discharge

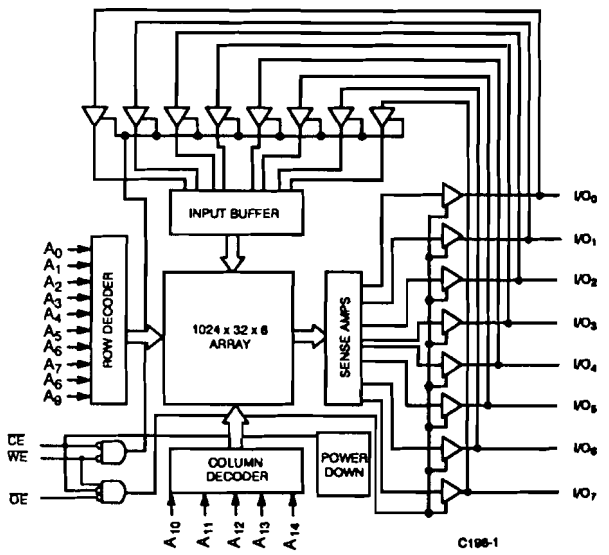
Functional Description

The CY7C198 and CY7C199 are high-performance CMOS static RAMs organized as 32,768 words by 8 bits. Easy memory expansion is provided by an active LOW chip enable ($\overline{CE}$) and active LOW output enable ($\overline{OE}$) and three-state drivers. Both devices have an automatic power-down feature, reducing the power consumption by 75% when deselected. The CY7C199 is in the space-saving 300-mil-wide DIP package and leadless chip carrier. The CY7C198 is in the standard 600-mil-wide package.

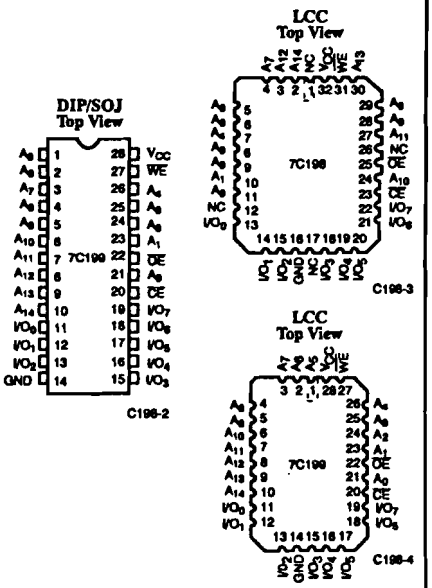
An active LOW write enable signal ($\overline{WE}$) controls the writing/reading operation of the memory. When $\overline{CE}$ and $\overline{WE}$ inputs are both LOW, data on the eight data input/output pins (I/O_0 through I/O_7) is written into the memory location addressed by the address present on the address pins (A_0 through A_{14}). Reading the device is accomplished by selecting the device and enabling the outputs, $\overline{CE}$ and $\overline{OE}$ active LOW, while $\overline{WE}$ remains inactive or HIGH. Under these conditions, the contents of the location addressed by the information on address pins is present on the eight data input/output pins.

The input/output pins remain in a high-impedance state unless the chip is selected, outputs are enabled, and write enable ($\overline{WE}$) is HIGH. A die coat is used to ensure alpha immunity.

Logic Block Diagram



Pin Configurations



Selection Guide

		7C198–12 7C199–12	7C198–15 7C199–15	7C198–20 7C199–20	7C198–25 7C199–25	7C198–35 7C199–35	7C198–45 7C199–45	7C198–55 7C199–55
Maximum Access Time (ns)		12	15	20	25	35	45	55
Maximum Operating Current (mA)	Commercial	160	160	160	160	150	150	150
	Military		180	180	160	160	160	160
Maximum Standby Current (mA)		40	40	40	35	35	35	35

Shaded area contains advanced information.

Maximum Ratings

(Above which the useful life may be impaired. For user guidelines, not tested.)

Storage Temperature ..... − 65°C to +150°C

Ambient Temperature with Power Applied ..... − 55°C to +125°C

Supply Voltage to Ground Potential (Pin 28 to Pin 14) ..... − 0.5V to +7.0V

DC Voltage Applied to Outputs in High Z State ..... − 0.5V to +7.0V

DC Input Voltage ..... − 3.0V to +7.0V

Output Current into Outputs (LOW) ..... 20 mA

Static Discharge Voltage ..... >2001V (per MIL-STD-883, Method 3015)

Latch-Up Current ..... >200 mA

Operating Range

Range	Ambient Temperature	V_{CC}
Commercial	0°C to +70°C	5V ± 10%
Military[1]	− 55°C to +125°C	5V ± 10%

Electrical Characteristics Over the Operating Range[2]

Parameters	Description	Test Conditions		7C198−12 7C199−12 Min.	7C198−12 7C199−12 Max.	7C198−15 7C199−15 Min.	7C198−15 7C199−15 Max.	7C198−20 7C199−20 Min.	7C198−20 7C199−20 Max.	Units
V_{OH}	Output HIGH Voltage	V_{CC} = Min., I_{OH} = − 4.0 mA		2.4		2.4		2.4		V
V_{OL}	Output LOW Voltage	V_{CC} = Min., I_{OL} = 8.0 mA			0.4		0.4		0.4	V
V_{IH}	Input HIGH Voltage			2.2	V_{CC}	2.2	V_{CC}	2.2	V_{CC}	V
V_{IL}	Input LOW Voltage			−0.5	0.8	−0.5	0.8	−0.5	0.8	V
I_{IX}	Input Load Current	GND ≤ V_I ≤ V_{CC}		−10	+10	−10	+10	−10	+10	μA
I_{OZ}	Output Leakage Current	GND ≤ V_I ≤ V_{CC}, Output Disabled		−10	+10	−10	+10	−10	+10	μA
I_{OS}	Output Short Circuit Current[3]	V_{CC} = Max., V_{OUT} = GND			−300		−300		−300	mA
I_{CC}	V_{CC} Operating Supply Current	V_{CC} = Max., I_{OUT} = 0 mA, $f = f_{MAX} = 1/t_{RC}$	Com'l		160		160		160	mA
			Mil				180		180	
I_{SB1}	Automatic CE Power-Down Current— TTL Inputs	Max. V_{CC}, $\overline{CE} \geq V_{IH}$, $V_{IN} \geq V_{IH}$ or $V_{IN} \leq V_{IL}$, $f = f_{MAX}$			40		40		40	mA
I_{SB2}	Automatic CE Power-Down Current— CMOS Inputs	Max. V_{CC}, $\overline{CE} \geq V_{CC}$ − 0.3V $V_{IN} \geq V_{CC}$ − 0.3V or $V_{IN} \leq$ 0.3V, f = 0			20		20		20	mA

Shaded area contains advanced information.

Notes:

1. T_A is the "instant on" case temperature.
2. See the last page of this specification for Group A subgroup testing information.
3. Not more than one output should be shorted at one time. Duration of the short circuit should not exceed 30 seconds.

Electrical Characteristics Over the Operating Range[2] (continued)

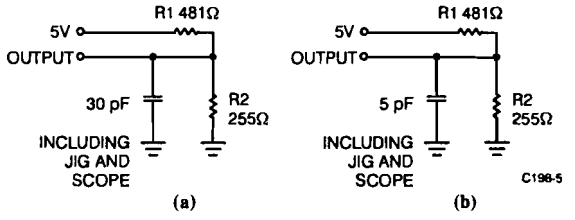
Parameters	Description	Test Conditions		7C198–25 7C199–25 Min.	7C198–25 7C199–25 Max.	7C198–35, 45, 55 7C199–35, 45, 55 Min.	7C198–35, 45, 55 7C199–35, 45, 55 Max.	Units
V_{OH}	Output HIGH Voltage	V_{CC} = Min., I_{OH} = – 4.0 mA		2.4		2.4		V
V_{OL}	Output LOW Voltage	V_{CC} = Min., I_{OL} = 8.0 mA			0.4		0.4	V
V_{IH}	Input HIGH Voltage			2.2	V_{CC}	2.2	V_{CC}	V
V_{IL}	Input LOW Voltage			–3.0	0.8	–3.0	0.8	V
I_{IX}	Input Load Current	GND ≤ V_I ≤ V_{CC}		–10	+10	–10	+10	μA
I_{OZ}	Output Leakage Current	GND ≤ V_I ≤ V_{CC}, Output Disabled		–10	+10	–10	+10	μA
I_{OS}	Output Short Circuit Current[3]	V_{CC} = Max., V_{OUT} = GND			–300		–300	mA
I_{CC}	V_{CC} Operating Supply Current	V_{CC} = Max., I_{OUT} = 0 mA, f = f_{MAX} = $1/t_{RC}$	Com'l		160		150	mA
			Mil		160		160	
I_{SB1}	Automatic CE Power-Down Current—TTL Inputs	Max. V_{CC}, $\overline{CE}$ ≥ V_{IH}, V_{IN} ≥ V_{IH} or V_{IN} ≤ V_{IL}, f = f_{MAX}			35		35	mA
I_{SB2}	Automatic CE Power-Down Current—CMOS Inputs	Max. V_{CC}, $\overline{CE}$ ≥ V_{CC} – 0.3V V_{IN} ≥ V_{CC} – 0.3V or V_{IN} ≤ 0.3V, f=0			20		20	mA

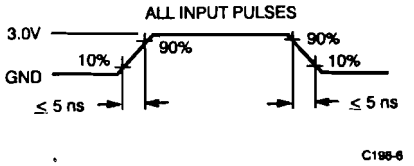
Capacitance[4]

Parameters	Description	Test Conditions	Max.	Units
C_{IN}	Input Capacitance	T_A = 25°C, f = 1 MHz, V_{CC} = 5.0V	10	pF
C_{OUT}	Output Capacitance		10	pF

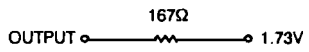
Note:
4. Tested initially and after any design or process changes that may affect these parameters.

AC Test Loads and Waveforms





Equivalent to: THÉVENIN EQUIVALENT



Switching Characteristics Over the Operating Range[2, 5]

Parameters	Description	7C198–12 7C199–12		7C198–15 7C199–15		7C198–20 7C199–20		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
READ CYCLE								
t_{RC}	Read Cycle Time	12		15		20		ns
t_{AA}	Address to Data Valid		12		15		20	ns
t_{OHA}	Data Hold from Address Change	3		3		3		ns
t_{ACE}	$\overline{CE}$ LOW to Data Valid		12		15		20	ns
t_{DOE}	$\overline{OE}$ LOW to Data Valid		6		8		10	ns
t_{LZOE}	$\overline{OE}$ LOW to Low Z[7]	0		0		0		ns
t_{HZOE}	$\overline{OE}$ HIGH to High Z[6, 7]		7		8		10	ns
t_{LZCE}	$\overline{CE}$ LOW to Low Z[7]	3		3		3		ns
t_{HZCE}	$\overline{CE}$ HIGH to High Z[6, 7]		7		8		10	ns
t_{PU}	$\overline{CE}$ LOW to Power-Up	0		0		0		ns
t_{PD}	$\overline{CE}$ HIGH to Power-Down		12		15		20	ns
WRITE CYCLE[8, 9]								
t_{WC}	Write Cycle Time	12		15		20		ns
t_{SCE}	$\overline{CE}$ LOW to Write End	9		10		15		ns
t_{AW}	Address Set-Up to Write End	9		10		15		ns
t_{HA}	Address Hold from Write End	0		0		0		ns
t_{SA}	Address Set-Up to Write Start	0		0		0		ns
t_{PWE}	$\overline{WE}$ Pulse Width	9		10		15		ns
t_{SD}	Data Set-Up to Write End	7		8		10		ns
t_{HD}	Data Hold from Write End	0		0		0		ns
t_{HZWE}	$\overline{WE}$ LOW to High Z[6]		7		7		10	ns
t_{LZWE}	$\overline{WE}$ HIGH to Low Z[7]	3		3		3		ns

Shaded area contains advanced information.

Notes:

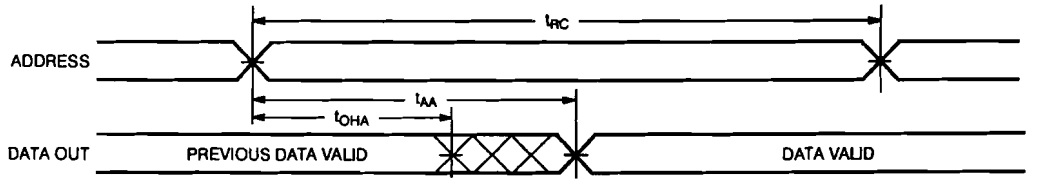
5. Test conditions assume signal transition time of 5 ns or less, timing reference levels of 1.5V, input pulse levels of 0 to 3.0V, and output loading of the specified I_{OL}/I_{OH} and 30-pF load capacitance.
6. t_{HZOE}, t_{HZCE}, and t_{HZWE} are specified with C_L = 5 pF as in part (b) of AC Test Loads. Transition is measured ±500 mV from steady state voltage.
7. At any given temperature and voltage condition, t_{HZCE} is less than t_{LZCE}, t_{HZOE} is less than t_{LZOE}, and t_{HZWE} is less than t_{LZWE} for any given device.
8. The internal write time of the memory is defined by the overlap of $\overline{CE}$ LOW and $\overline{WE}$ LOW. Both signals must be LOW to initiate a write and either signal can terminate a write by going HIGH. The data input set-up and hold timing should be referenced to the rising edge of the signal that terminates the write.
9. The minimum write cycle time for write cycle #3 ($\overline{WE}$ controlled, $\overline{OE}$ LOW) is the sum of t_{HZWE} and t_{SD}.

Switching Characteristics Over the Operating Range[2, 5] (continued)

Parameters	Description	7C198–25 7C199–25		7C198–35 7C199–35		7C198–45 7C199–45		7C198–55 7C199–55		Units
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
READ CYCLE										
t_{RC}	Read Cycle Time	25		35		45		55		ns
t_{AA}	Address to Data Valid		25		35		45		55	ns
t_{OHA}	Data Hold from Address Change	3		3		3		3		ns
t_{ACE}	$\overline{CE}$ LOW to Data Valid		25		35		45		55	ns
t_{DOE}	$\overline{OE}$ LOW to Data Valid		15		20		20		20	ns
t_{LZOE}	$\overline{OE}$ LOW to Low Z[7]	3		3		3		3		ns
t_{HZOE}	$\overline{OE}$ HIGH to High Z[6, 7]		13		15		20		25	ns
t_{LZCE}	$\overline{CE}$ LOW to Low Z[7]	3		3		3		3		ns
t_{HZCE}	$\overline{CE}$ HIGH to High Z[6,7]		13		15		20		25	ns
t_{PU}	$\overline{CE}$ LOW to Power-Up	0		0		0		0		ns
t_{PD}	$\overline{CE}$ HIGH to Power-Down		20		20		25		25	ns
WRITE CYCLE[8, 9]										
t_{WC}	Write Cycle Time	25		35		45		50		ns
t_{SCE}	$\overline{CE}$ LOW to Write End	20		30		40		50		ns
t_{AW}	Address Set-Up to Write End	20		30		40		50		ns
t_{HA}	Address Hold from Write End	0		0		0		0		ns
t_{SA}	Address Set-Up to Write Start	0		0		0		0		ns
t_{PWE}	$\overline{WE}$ Pulse Width	20		25		30		40		ns
t_{SD}	Data Set-Up to Write End	15		17		20		25		ns
t_{HD}	Data Hold from Write End	0		0		0		0		ns
t_{HZWE}	$\overline{WE}$ LOW to High Z[6]		13		15		20		25	ns
t_{LZWE}	$\overline{WE}$ HIGH to Low Z[7]	3		3		3		3		ns

Switching Waveforms

Read Cycle No. 1[10, 11]

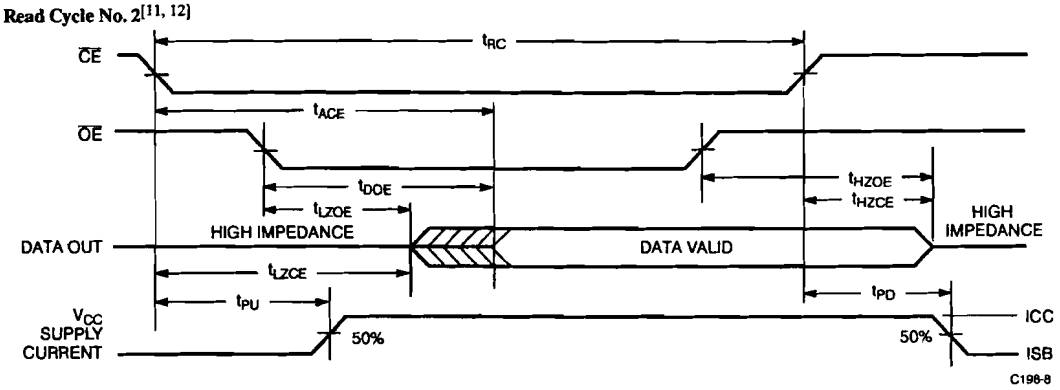


C198-7

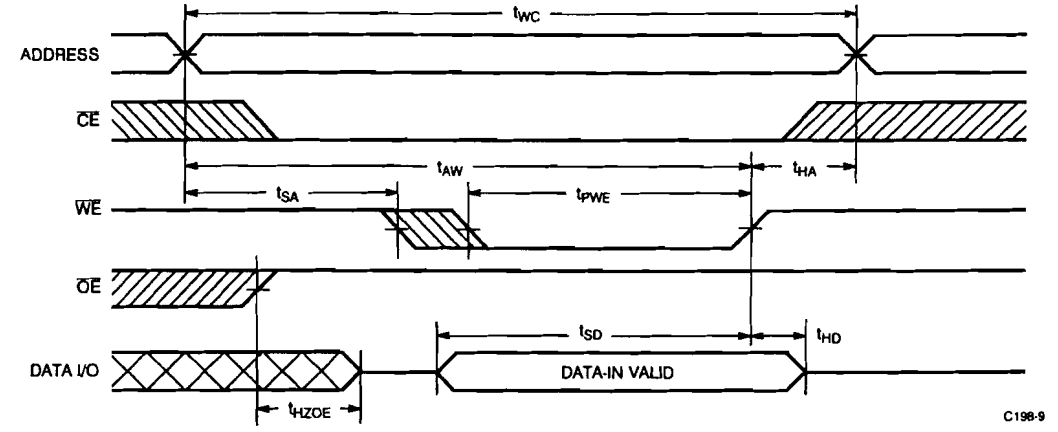
Notes:
10. Device is continuously selected. $\overline{OE}$, $\overline{CE}$ = V_{IL}.
11. $\overline{WE}$ is HIGH for read cycle.

Switching Waveforms (continued)

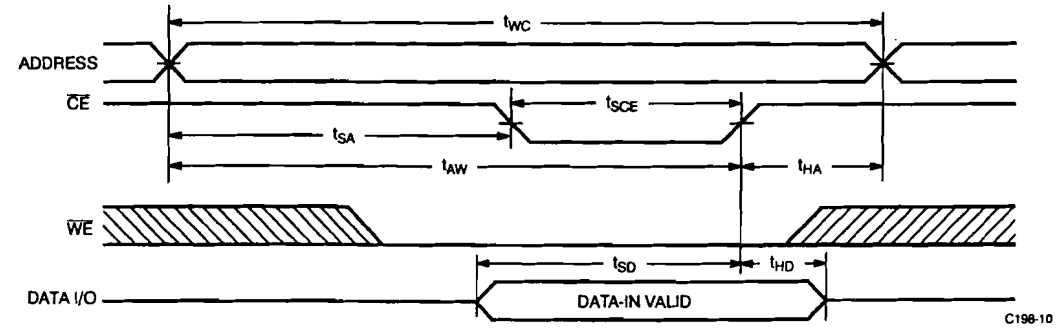
Read Cycle No. 2[11, 12]



Write Cycle No. 1 ($\overline{WE}$ Controlled)[8, 13, 14]



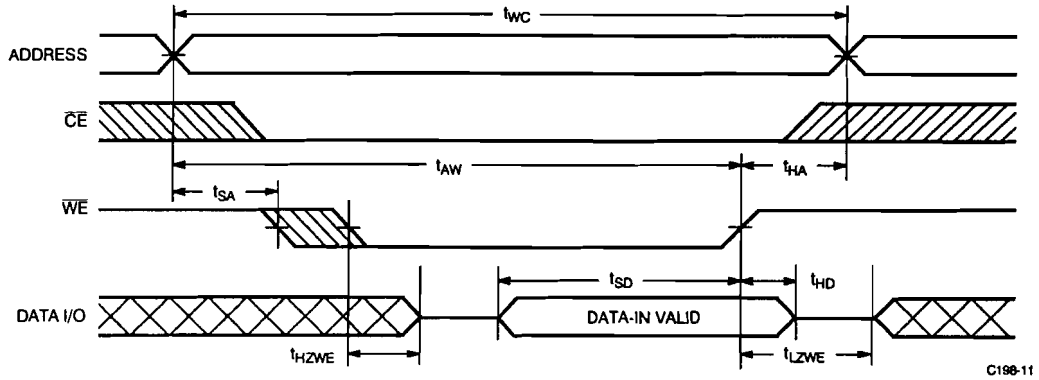
Write Cycle No. 2 ($\overline{CE}$ Controlled)[8, 13, 14]



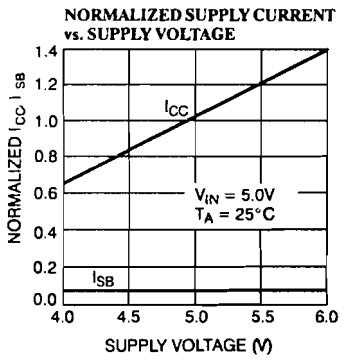
Notes:
12. Address valid prior to or coincident with $\overline{CE}$ transition LOW.
13. Data I/O is high impedance if $\overline{OE}$ = V_{IH}.
14. If $\overline{CE}$ goes HIGH simultaneously with $\overline{WE}$ HIGH, the output remains in a high-impedance state.

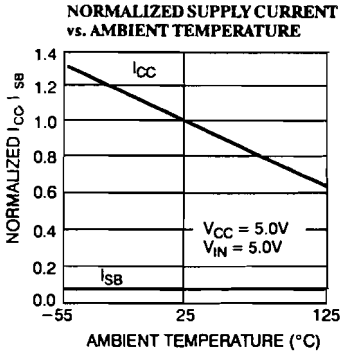
Switching Waveforms (continued)

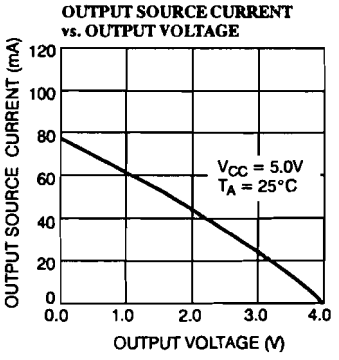
Write Cycle No. 3 ($\overline{WE}$ Controlled, $\overline{OE}$ LOW)[9, 14]

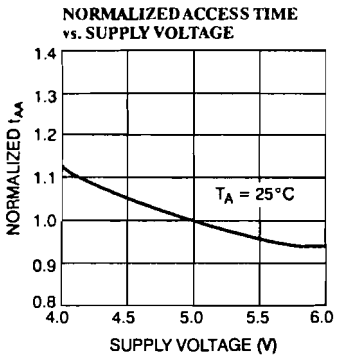


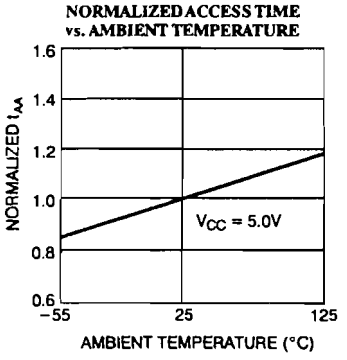
Typical DC and AC Characteristics

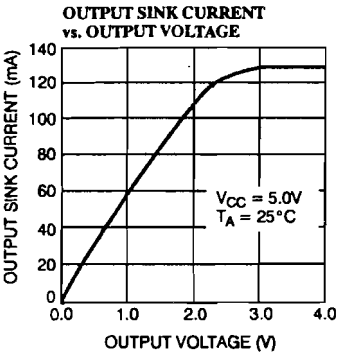






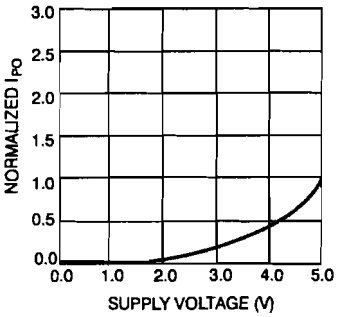




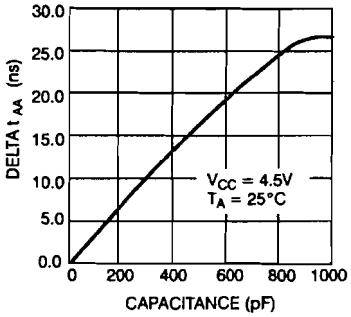


Typical DC and AC Characteristics (continued)

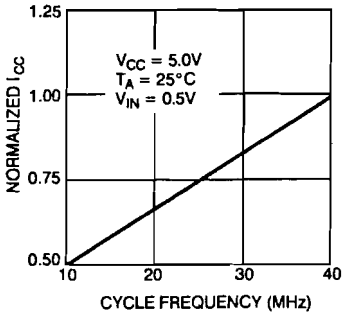
TYPICAL POWER-ON CURRENT vs. SUPPLY VOLTAGE



TYPICAL ACCESS TIME CHANGE vs. OUTPUT LOADING



NORMALIZED I_{CC} vs. CYCLE TIME



Truth Table

$\overline{CE}$	$\overline{WE}$	$\overline{OE}$	Inputs/Outputs	Mode	Power
H	X	X	High Z	Deselect/Power-Down	Standby (I_{SB})
L	H	L	Data Out	Read	Active (I_{CC})
L	L	X	Data In	Write	Active (I_{CC})
L	H	H	High Z	Deselect, Output Disabled	Active (I_{CC})

National Semiconductor

NMC27C256
262,144-Bit (32k x 8) UV Erasable CMOS PROM

General Description

The NMC27C256 is a high-speed 256k UV erasable and electrically reprogrammable CMOS EPROM, ideally suited for applications where fast turnaround, pattern experimentation and low power consumption are important requirements.

The NMC27C256 is designed to operate with a single +5V power supply with ±5% or ±10% tolerance. The CMOS design allows the part to operate over extended and military temperature ranges.

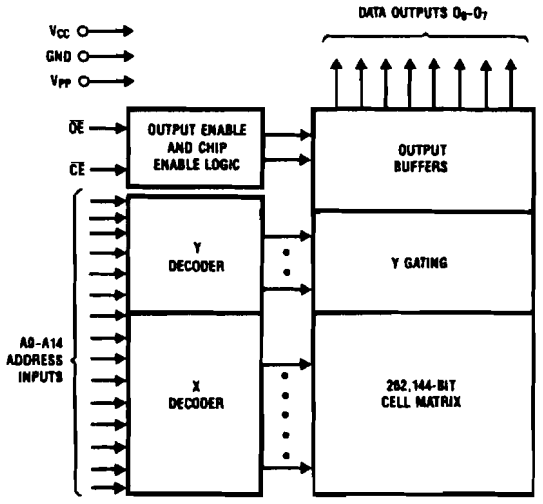
The NMC27C256 is packaged in a 28-pin dual in-line package with transparent lid. The transparent lid allows the user to expose the chip to ultraviolet light to erase the bit pattern. A new pattern can then be written electrically into the device by following the programming procedure.

This EPROM is fabricated with National's proprietary, time proven CMOS double-poly silicon gate technology which combines high performance and high density with low power consumption and excellent reliability.

Features

- Clocked sense amps for fast access time down to 170 ns
- Low CMOS power consumption
  - Active power: 55 mW max
  - Standby power: 0.55 mW max
- Performance compatible to NSC800™ CMOS microprocessor
- Single 5V power supply
- Extended temperature range (NMC27C256QE), −40°C to +85°C, and military temperature range (NMC27C256QM), −55°C to +125°C, available
- Pin compatible with NMOS 256k EPROMs
- Fast and reliable programming (0.5 ms for most bytes)
- Static operation—no clocks required
- TTL, CMOS compatible inputs/outputs
- TRI-STATE® output
- Optimum EPROM for total CMOS systems

Block Diagram



TL/D/7512-1

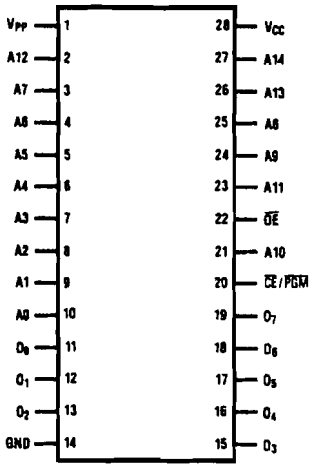
Pin Names

A0–A14	Addresses
$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
O_0–O_7	Outputs
$\overline{PGM}$	Program
NC	No Connect

Connection Diagram

27C512 27512	27C128 27128	27C64 2764	27C32 2732	27C16 2716
A15	V_{PP}	V_{PP}		
A12	A12	A12		
A7	A7	A7	A7	A7
A6	A6	A6	A6	A6
A5	A5	A5	A5	A5
A4	A4	A4	A4	A4
A3	A3	A3	A3	A3
A2	A2	A2	A2	A2
A1	A1	A1	A1	A1
A0	A0	A0	A0	A0
O_0	O_0	O_0	O_0	O_0
O_1	O_1	O_1	O_1	O_1
O_2	O_2	O_2	O_2	O_2
GND	GND	GND	GND	GND

NMC27C256Q
Dual-In-Line Package



TL/D/7512-2

27C16 2718	27C32 2732	27C64 2764	27C128 27128	27C512 27512
		V_{CC}	V_{CC}	V_{CC}
		$\overline{PGM}$	$\overline{PGM}$	A14
V_{CC}	V_{CC}	NC	A13	A13
A8	A8	A8	A8	A8
A9	A9	A9	A9	A9
V_{PP}	A11	A11	A11	A11
$\overline{OE}$	$\overline{OE}/V_{PP}$	$\overline{OE}$	$\overline{OE}$	$\overline{OE}/V_{PP}$
A10	A10	A10	A10	A10
$\overline{CE}/\overline{PGM}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$	$\overline{CE}$
O_7	O_7	O_7	O_7	O_7
O_6	O_6	O_6	O_6	O_6
O_5	O_5	O_5	O_5	O_5
O_4	O_4	O_4	O_4	O_4
O_3	O_3	O_3	O_3	O_3

Note: Socket compatible EPROM pin configurations are shown in the blocks adjacent to the NMC27C256 pins.

Order Number NMC27C256Q
See NS Package Number J28AQ

Commercial Temp Range (0°C to +70°C)
V_{CC} = 5V ±5%

Parameter/Order Number	Access Time
NMC27C256Q17	170
NMC27C256Q20	200
NMC27C256Q25	250

Commercial Temp Range (0°C to +70°C)
V_{CC} = 5V ±10%

Parameter/Order Number	Access Time
NMC27C256Q200	200
NMC27C256Q250	250
NMC27C256Q300	300

Extended Temp Range (−40°C to +85°C)
V_{CC} = 5V ±10%

Parameter/Order Number	Access Time
NMC27C256QE200	200
NMC27C256QE250	250

Military Temp Range (−55°C to +125°C)
V_{CC} = 5V ±10%

Parameter/Order Number	Access Time
NMC27C256QM250	250
NMC27C256QM350	350

NOTE: For plastic DIP and surface mount PLCC package requirements please refer to NMC27C256BN data sheet.

COMMERCIAL TEMPERATURE RANGE

Absolute Maximum Ratings (Note 1)

Temperature Under Bias	−10°C to +80°C
Storage Temperature	−65°C to +150°C
All Input Voltages with Respect to Ground (Note 10)	+6.5V to −0.6V
All Output Voltages with Respect to Ground (Note 10)	V_{CC} + 1.0V to GND −0.6V
V_{PP} Supply Voltage with Respect to Ground During Programming	+14.0V to −0.6V
Power Dissipation	1.0W
Lead Temperature (Soldering, 10 sec.)	300°C
V_{CC} Supply Voltage with Respect to Ground	+7.0V to −0.6V

Operating Conditions (Note 7)

Temperature Range	0°C to +70°C
V_{CC} Power Supply	
NMC27C256Q17, 20, 25	5V ±5%
NMC27C256Q200, 250, 300	5V ±10%

READ OPERATION

DC Electrical Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
I_{LI}	Input Load Current	$V_{IN} = V_{CC}$ or GND			10	μA
I_{LO}	Output Leakage Current	$V_{OUT} = V_{CC}$ or GND, $\overline{CE} = V_{IH}$			10	μA
I_{CC1} (Note 9)	V_{CC} Current (Active) TTL Inputs	$\overline{CE} = V_{IL}$, f = 5 MHz Inputs = V_{IH} or V_{IL}, I/O = 0 mA		6	20	mA
I_{CC2} (Note 9)	V_{CC} Current (Active) CMOS Inputs	$\overline{CE}$ = GND, f = 5 MHz Inputs = V_{CC} or GND, I/O = 0 mA		3	10	mA
I_{CCSB1}	V_{CC} Current (Standby) TTL Inputs	$\overline{CE} = V_{IH}$		0.1	1	mA
I_{CCSB2}	V_{CC} Current (Standby) CMOS Inputs	$\overline{CE} = V_{CC}$		0.5	100	μA
I_{PP}	V_{PP} Load Current	$V_{PP} = V_{CC}$			10	μA
V_{IL}	Input Low Voltage		−0.1		0.8	V
V_{IH}	Input High Voltage		2.0		V_{CC} + 1	V
V_{OL1}	Output Low Voltage	I_{OL} = 2.1 mA			0.45	V
V_{OH1}	Output High Voltage	I_{OH} = −400 μA	2.4			V
V_{OL2}	Output Low Voltage	I_{OL} = 0 μA			0.1	V
V_{OH2}	Output High Voltage	I_{OH} = 0 μA	V_{CC} − 0.1			V

AC Electrical Characteristics

Symbol	Parameter	Conditions	NMC27C256								Units
			Q17		Q20, Q200		Q25, Q250		Q300		
			Min	Max	Min	Max	Min	Max	Min	Max	
t_{ACC}	Address to Output Delay	$\overline{CE} = \overline{OE} = V_{IL}$		170		200		250		300	ns
t_{CE}	$\overline{CE}$ to Output Delay	$\overline{OE} = V_{IL}$		170		200		250		300	ns
t_{OE}	$\overline{OE}$ to Output Delay	$\overline{CE} = V_{IL}$		75		75		100		120	ns
t_{DF}	$\overline{OE}$ High to Output Float	$\overline{CE} = V_{IL}$	0	60	0	60	0	60	0	105	ns
t_{CF}	$\overline{CE}$ High to Output Float	$\overline{OE} = V_{IL}$	0	60	0	60	0	60	0	105	ns
t_{OH}	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$, Whichever Occurred First	$\overline{CE} = \overline{OE} = V_{IL}$	0		0		0		0		ns

MILITARY AND EXTENDED TEMPERATURE RANGE

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Temperature Under Bias	Operating Temp Range
Storage Temperature	−65°C to +150°C
All Input Voltages with Respect to Ground (Note 10)	+6.5V to −0.6V
All Output Voltages with Respect to Ground (Note 10)	V_{CC} + 1.0V to GND − 0.6V
V_{PP} Supply Voltage with Respect to Ground During Programming	+14.0V to −0.6V
Power Dissipation	1.0W
Lead Temperature (Soldering, 10 sec.)	300°C
V_{CC} Supply Voltage with Respect to Ground	+7.0V to −0.6V

Operating Conditions (Note 7)

Temperature Range	
NMC27C256QE200, 250	−40°C to +85°C
NMC27C256QM250, M350	−55°C to +125°C
V_{CC} Power Supply	5V ±10%

READ OPERATION

DC Electrical Characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Units
I_{LI}	Input Load Current	$V_{IN} = V_{CC}$ or GND			10	μA
I_{LO}	Output Leakage Current	$V_{OUT} = V_{CC}$ or GND, $\overline{CE} = V_{IH}$			10	μA
I_{CC1} (Note 9)	V_{CC} Current (Active) TTL Inputs	$\overline{CE} = V_{IL}$, f = 5 MHz Inputs = V_{IH} or V_{IL}, I/O = 0 mA		6	20	mA
I_{CC2} (Note 9)	V_{CC} Current (Active) CMOS Inputs	$\overline{CE}$ = GND, f = 5 MHz Inputs = V_{CC} or GND, I/O = 0 mA		3	10	mA
I_{CCSB1}	V_{CC} Current (Standby) TTL Inputs	$\overline{CE} = V_{IH}$		0.1	1	mA
I_{CCSB2}	V_{CC} Current (Standby) CMOS Inputs	$\overline{CE} = V_{CC}$		0.5	100	μA
I_{PP}	V_{PP} Load Current	$V_{PP} = V_{CC}$			10	μA
V_{IL}	Input Low Voltage		−0.1		0.8	V
V_{IH}	Input High Voltage		2.0		V_{CC} + 1	V
V_{OL1}	Output Low Voltage	I_{OL} = 2.1 mA			0.45	V
V_{OH1}	Output High Voltage	I_{OH} = −400 μA	2.4			V
V_{OL2}	Output Low Voltage	I_{OH} = 0 μA			0.1	V
V_{OH2}	Output High Voltage	I_{OH} = 0 μA	V_{CC} − 0.1			V

AC Electrical Characteristics

Symbol	Parameter	Conditions	NMC27C256Q						Units
			E200		E250 M250		M350		
			Min	Max	Min	Max	Min	Max	
t_{ACC}	Address to Output Delay	$\overline{CE} = \overline{OE} = V_{IL}$		200		250		350	ns
t_{CE}	$\overline{CE}$ to Output Delay	$\overline{OE} = V_{IL}$		200		250		350	ns
t_{OE}	$\overline{OE}$ to Output Delay	$\overline{CE} = V_{IL}$		75		100		120	ns
t_{DF}	$\overline{OE}$ High to Output Float	$\overline{CE} = V_{IL}$	0	60	0	60	0	105	ns
t_{OH}	Output Hold from Addresses, $\overline{CE}$ or $\overline{OE}$ Whichever Occurred First	$\overline{CE} = \overline{OE} = V_{IL}$	0		0		0		ns
t_{CF}	$\overline{CE}$ High to Output Float	$\overline{OE} = V_{IL}$	0	60	0	60	0	105	ns

Capacitance T_A = +25°C, f = 1 MHz (Note 2)

Symbol	Parameter	Conditions	Typ	Max	Units
C_{IN}	Input Capacitance	V_{IN} = 0V	6	12	pF
C_{OUT}	Output Capacitance	V_{OUT} = 0V	9	12	pF

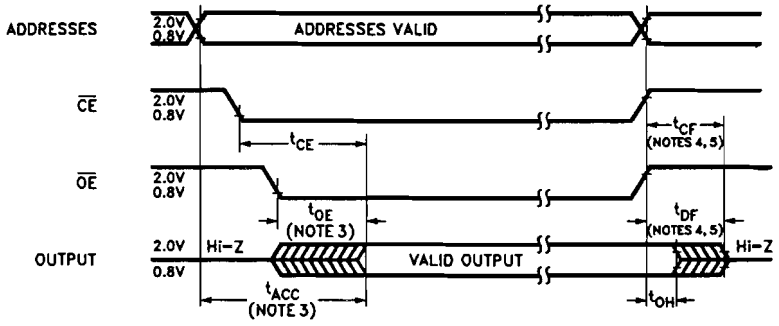
AC Test Conditions

Output Load	1 TTL Gate and C_L = 100 pF (Note 8)
Input Rise and Fall Times	≤5 ns
Input Pulse Levels	0.45V to 2.4V

Timing Measurement Reference Level

Inputs	0.8V and 2V
Outputs	0.8V and 2V

AC Waveforms (Notes 6, 7 & 9)



TL/D/7512-3

Note 1: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Note 2: This parameter is only sampled and is not 100% tested.

Note 3: $\overline{OE}$ may be delayed up to $t_{ACC} - t_{OE}$ after the falling edge of $\overline{CE}$ without impacting t_{ACC}.

Note 4: The t_{DF} and t_{CF} compare level is determined as follows:
High to TRI-STATE, the measured V_{OH1} (DC) − 0.10V;
Low to TRI-STATE, the measured V_{OL1} (DC) + 0.10V.

Note 5: TRI-STATE may be attained using $\overline{OE}$ or $\overline{CE}$.

Note 6: The power switching characteristics of EPROMs require careful device decoupling. It is recommended that at least a 0.1 μF ceramic capacitor be used on every device between V_{CC} and GND.

Note 7: The outputs must be restricted to V_{CC} + 1.0V to avoid latch-up and device damage.

Note 8: 1 TTL Gate: I_{OL} = 1.6 mA, I_{OH} = −400 μA.
C_L: 100 pF includes fixture capacitance.

Note 9: V_{PP} may be connected to V_{CC} except during programming.

Note 10: Inputs and outputs can undershoot to −2.0V for 20 ns Max.

Programming Characteristics (Notes 1, 2, 3 & 4)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{AS}	Address Setup Time		2			μs
t_{OES}	$\overline{OE}$ Setup Time		2			μs
t_{VPS}	V_{PP} Setup Time		2			μs
t_{VCS}	V_{CC} Setup Time		2			μs
t_{DS}	Data Setup Time		2			μs
t_{AH}	Address Hold Time		0			μs
t_{DH}	Data Hold Time		2			μs
t_{DF}	Output Enable to Output Float Delay	$\overline{CE} = V_{IL}$	0		130	ns
t_{PW}	Program Pulse Width		0.5	0.5	10	ms
t_{OE}	Data Valid from $\overline{OE}$	$\overline{CE} = V_{IL}$			150	ns
I_{PP}	V_{PP} Supply Current During Programming Pulse	$\overline{CE} = V_{IL}$ $\overline{PGM} = V_{IL}$			30	mA
I_{CC}	V_{CC} Supply Current				10	mA
T_A	Temperature Ambient		20	25	30	°C
V_{CC}	Power Supply Voltage		5.75	6.0	6.25	V
V_{PP}	Programming Supply Voltage		12.2	13.0	13.3	V
t_{FR}	Input Rise, Fall Time		5			ns
V_{IL}	Input Low Voltage			0.0	0.45	V
V_{IH}	Input High Voltage		2.4	4.0		V
t_{IN}	Input Timing Reference Voltage		0.8	1.5	2.0	V
t_{OUT}	Output Timing Reference Voltage		0.8	1.5	2.0	V

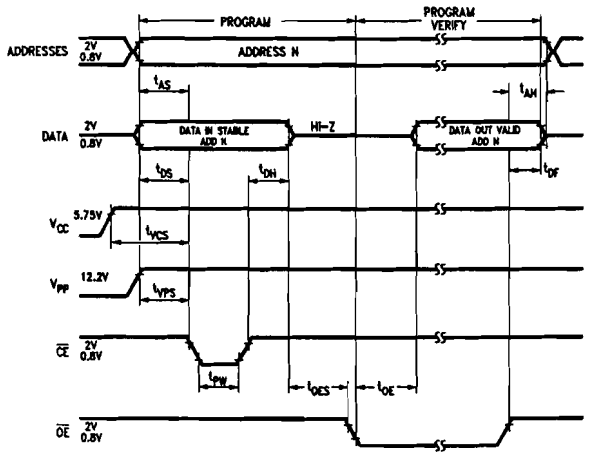
Note 1: National's standard product warranty applies only to devices programmed to specifications described herein.

Note 2: V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}. The EPROM must not be inserted into or removed from a board with voltage applied to V_{PP} or V_{CC}.

Note 3: The maximum absolute allowable voltage which may be applied to the V_{PP} pin during programming is 14V. Care must be taken when switching the V_{PP} supply to prevent any overshoot from exceeding this 14V maximum specification. At least a 0.1 μF capacitor is required across V_{PP}, V_{CC} to GND to suppress spurious voltage transients which may damage the device.

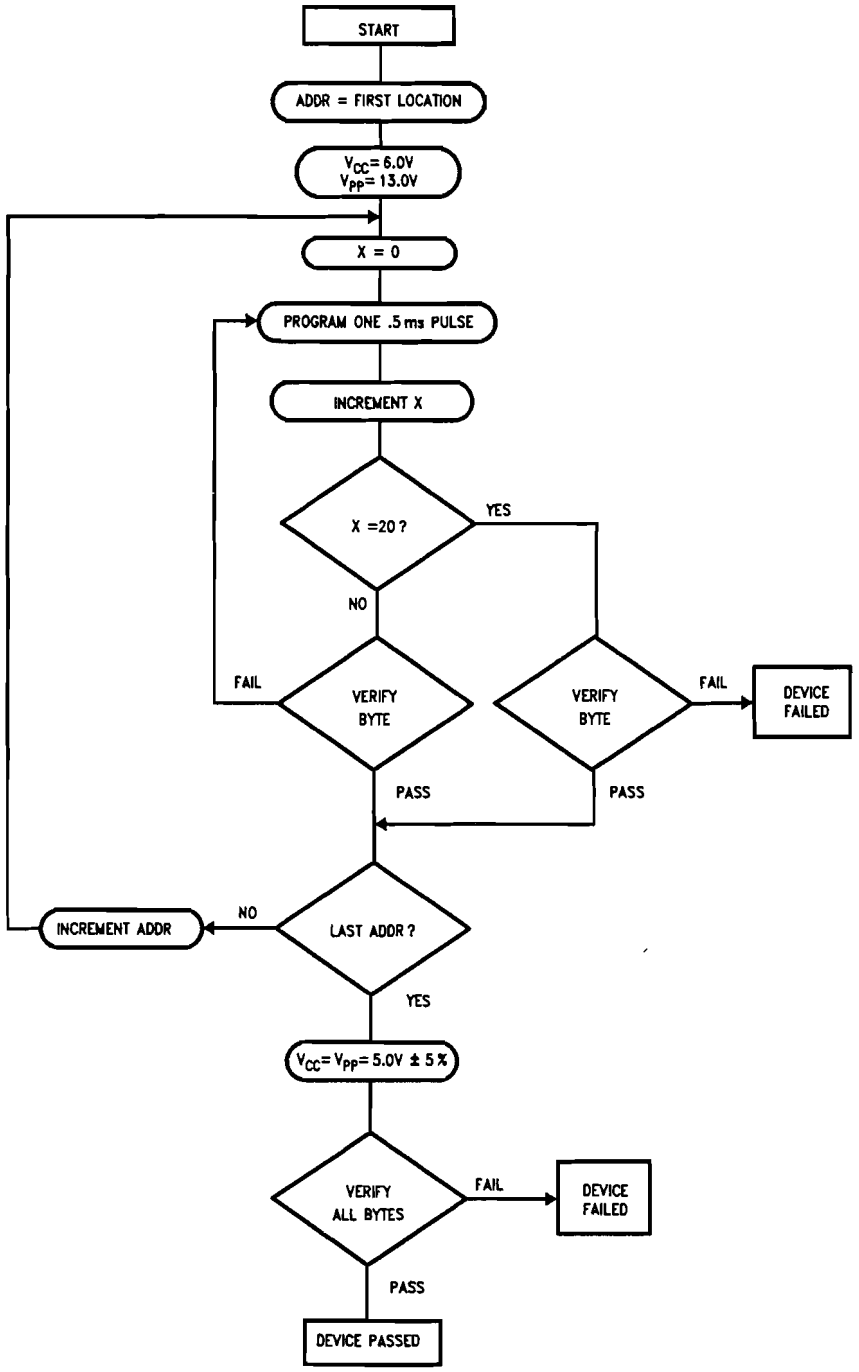
Note 4: Programming and program verify are tested with the Interactive Program Algorithm, at typical power supply voltages and timings.

Programming Waveforms (Note 3)



TL/D/7512-4

Interactive Programming Algorithm Flow Chart



TL/D/7512-5

FIGURE 1

Functional Description

DEVICE OPERATION

The six modes of operation of the NMC27C256 are listed in Table I. It should be noted that all inputs for the six modes are at TTL levels. The power supplies required are V_{CC} and V_{PP}. The V_{PP} power supply must be at 13.0V during the three programming modes, and must be at 5V in the other three modes. The V_{CC} power supply must be at 6V during the three programming modes, and at 5V in the other three modes.

Read Mode

The NMC27C256 has two control functions, both of which must be logically active in order to obtain data at the outputs. Chip Enable ($\overline{CE}$) is the power control and should be used for device selection. Output Enable ($\overline{OE}$) is the output control and should be used to gate data to the output pins, independent of device selection. Assuming that addresses are stable, address access time (t_{ACC}) is equal to the delay from $\overline{CE}$ to output (t_{CE}). Data is available at the outputs t_{OE} after the falling edge of $\overline{OE}$, assuming that $\overline{CE}$ has been low and addresses have been stable for at least $t_{ACC} - t_{OE}$.

The sense amps are clocked for fast access time. V_{CC} should therefore be maintained at operating voltage during read and verify. If V_{CC} temporarily drops below the spec. voltage (but not to ground) an address transition must be performed after the drop to ensure proper output data.

Standby Mode

The NMC27C256 has a standby mode which reduces the active power dissipation by 99%, from 55 mW to 0.55 mW. The NMC27C256 is placed in the standby mode by applying a CMOS high signal to the $\overline{CE}$ input. When in standby mode, the outputs are in a high impedance state, independent of the $\overline{OE}$ input.

Output OR-Tying

Because NMC27C256s are usually used in larger memory arrays, National has provided a 2-line control function that accommodates this use of multiple memory connections. The 2-line control function allows for:

a) the lowest possible memory power dissipation, and

b) complete assurance that output bus contention will not occur.

To most efficiently use these two control lines, it is recommended that $\overline{CE}$ (pin 20) be decoded and used as the primary device selecting function, while $\overline{OE}$ (pin 22) be made a common connection to all devices in the array and connected to the READ line from the system control bus. This assures that all deselected memory devices are in their low power standby modes and that the output pins are active only when data is desired from a particular memory device.

Programming

CAUTION: Exceeding 14V on pin 1 (V_{PP}) will damage the NMC27C256.

Initially, and after each erasure, all bits of the NMC27C256 are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be presented in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The NMC27C256 is in the programming mode when the V_{PP} power supply is at 13.0V and $\overline{OE}$ is at V_{IH}. It is required that at least a 0.1 μF capacitor be placed across V_{PP}, V_{CC} to ground to suppress spurious voltage transients which may damage the device. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

When the address and data are stable, an active low TTL program pulse is applied to the $\overline{CE}/\overline{PGM}$ input. A program pulse must be applied at each address location to be programmed. Any location may be programmed at any time—either individually, sequentially, or at random. The NMC27C256 is designed to be programmed with interactive programming, where each address is programmed with a series of 0.5 ms pulses until it verifies (up to a maximum of 20 pulses or 10 ms). The NMC27C256 must not be programmed with a DC signal applied to the $\overline{CE}/\overline{PGM}$ input.

Programming multiple NMC27C256s in parallel with the same data can be easily accomplished due to the simplicity of the programming requirements. Like inputs of the paralleled NMC27C256s may be connected together when they are programmed with the same data. A low level TTL pulse applied to the $\overline{CE}/\overline{PGM}$ input programs the paralleled NMC27C256s.

TABLE I. Mode Selection

Mode \ Pins	$\overline{CE}/\overline{PGM}$ (20)	$\overline{OE}$ (22)	V_{PP} (1)	V_{CC} (28)	Outputs (11–13, 15–19)
Read	V_{IL}	V_{IL}	5V	5V	D_{OUT}
Standby	V_{IH}	Don't Care	5V	5V	Hi-Z
Program	V_{IL}	V_{IH}	13.0V	6V	D_{IN}
Program Verify	V_{IH}	V_{IL}	13.0V	6V	D_{OUT}
Program Inhibit	V_{IH}	V_{IH}	13.0V	6V	Hi-Z
Output Disable	Don't Care	V_{IH}	5V	5V	Hi-Z

Functional Description (Continued)

Program Inhibit

Programming multiple NMC27C256s in parallel with different data is also easily accomplished. Except for $\overline{CE}$ all like inputs (including $\overline{OE}$) of the parallel NMC27C256s may be common. A TTL low level program pulse applied to an NMC27C256's $\overline{CE}/\overline{PGM}$ input with V_{PP} at 13.0V will program that NMC27C256. A TTL high level $\overline{CE}$ input inhibits the other NMC27C256s from being programmed.

Program Verify

A verify should be performed on the programmed bits to determine whether they were correctly programmed. The verify may be performed with V_{PP} at 13.0V. V_{PP} must be at V_{CC}, except during programming and program verify.

ERASURE CHARACTERISTICS

The erasure characteristics of the NMC27C256 are such that erasure begins to occur when exposed to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å–4000Å range.

After programming, opaque labels should be placed over the NMC27C256's window to prevent unintentional erasure. Covering the window will also prevent temporary functional failure due to the generation of photo currents.

The recommended erasure procedure for the NMC27C256 is exposure to short wave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of 15W-sec/cm^2.

The NMC27C256 should be placed within 1 inch of the lamp tubes during erasure. Some lamps have a filter on their tubes which should be removed before erasure. Table II shows the minimum NMC27C256 erasure time for various light intensities.

An erasure system should be calibrated periodically. The distance from lamp to unit should be maintained at one inch. The erasure time increases as the square of the distance. (If distance is doubled the erasure time increases by a factor of 4.) Lamps lose intensity as they age. When a lamp is changed, the distance has changed or the lamp has aged, the system should be checked to make certain full erasure is occurring. Incomplete erasure will cause symptoms that can be misleading. Programmers, components, and even system designs have been erroneously suspected when incomplete erasure was the problem.

SYSTEM CONSIDERATION

The power switching characteristics of EPROMs require careful decoupling of the devices. The supply current, I_{CC}, has three segments that are of interest to the system designer—the standby current level, the active current level, and the transient current peaks that are produced by voltage transitions on input pins. The magnitude of these transient current peaks is dependent on the output capacitance loading of the device. The associated V_{CC} transient voltage peaks can be suppressed by properly selected decoupling capacitors. It is recommended that at least a 0.1 μF ceramic capacitor be used on every device between V_{CC} and GND. This should be a high frequency capacitor of low inherent inductance. In addition, at least a 4.7 μF bulk electrolytic capacitor should be used between V_{CC} and GND for each eight devices. The bulk capacitor should be located near where the power supply is connected to the array. The purpose of the bulk capacitor is to overcome the voltage drop caused by the inductive effects of the PC board traces.

TABLE II. Minimum NMC27C256 Erasure Time

Light Intensity (Micro-Watts/cm^2)	Erasure Time (Minutes)
15,000	20
10,000	25
5,000	50

ANALOG DEVICES

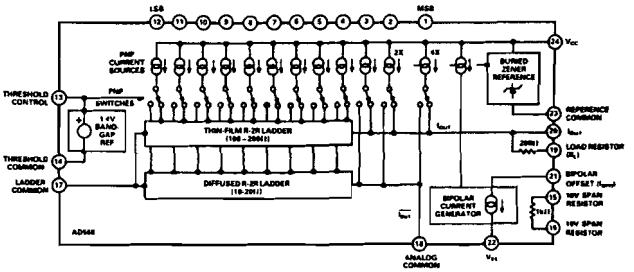
12-Bit Ultrahigh Speed Monolithic D/A Converter

AD568

FEATURES

- Ultrahigh Speed: Current Settling to 1LSB in 35ns
- High Stability Buried Zener Reference on Chip
- Monotonicity Guaranteed Over Temperature
- 10.24mA Full-Scale Output Suitable for Video Applications
- Integral and Differential Linearity Guaranteed Over Temperature
- 0.3" "Skinny DIP" Packaging
- Variable Threshold Allows TTL and CMOS Interface
- MIL-STD-883 Compliant Versions Available

FUNCTIONAL BLOCK DIAGRAM



PRODUCT DESCRIPTION

The AD568 is an ultrahigh-speed, 12-bit digital-to-analog converter (DAC) settling to 0.025% in 35ns. The monolithic device is fabricated using Analog Devices' Complementary Bipolar (CB) Process. This is a proprietary process featuring high-speed NPN and PNP devices on the same chip without the use of dielectric isolation or multichip hybrid techniques. The high speed of the AD568 is maintained by keeping impedance levels low enough to minimize the effects of parasitic circuit capacitances.

The DAC consists of 16 current sources configured to deliver a 10.24mA full-scale current. Multiple matched current sources and thin-film ladder techniques are combined to produce bit weighting. The DAC's output is a 10.24mA full scale (FS) for current output applications or a 1.024V FS unbuffered voltage output. Additionally, a 10.24V FS buffered output may be generated using an onboard 1kΩ span resistor with an external op amp. Bipolar ranges are accomplished by pin strapping.

Laser wafer trimming insures full 12-bit linearity. All grades of the AD568 are guaranteed monotonic over their full operating temperature range. Furthermore, the output resistance of the DAC is trimmed to 100Ω ± 1.0%. The gain temperature coefficient of the voltage output is 30ppm/°C max (K).

The AD568 is available in three performance grades. The AD568JQ and KQ are available in 24-pin cerdip (0.3") packages and are specified for operation from 0 to +70°C. The AD568SQ features operation from −55°C to +125°C and is also packaged in the hermetic 0.3" cerdip.

PRODUCT HIGHLIGHTS

1. The ultrafast settling time of the AD568 allows leading edge performance in waveform generation, graphics display and high-speed A/D conversion applications.
2. Pin strapping provides a variety of voltage and current output ranges for application versatility. Tight control of the absolute output current reduces trim requirements in externally-scaled applications.
3. Matched on-chip resistors can be used for precision scaling in high-speed A/D conversion circuits.
4. The digital inputs are compatible with TTL and +5V CMOS logic families.
5. Skinny DIP (0.3") packaging minimizes board space requirements and eases layout considerations.
6. The AD568 is available in versions compliant with MIL-STD-883. Refer to the Analog Devices Military Products Databook or current AD568/883B data sheet for detailed specifications.

AD568—SPECIFICATIONS (@ = +25°C, V_{CC}, V_{EE} = ±15V unless otherwise noted.)

Model	AD568J			AD568K			AD568S			Units
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
RESOLUTION	12			12			12			Bits
ACCURACY[1]										
Linearity	−1/2		+1/2	−1/4		+1/4	−1/2		+1/2	LSB
T_{min} to T_{max}	−3/4		+3/4	−1/2		+1/2	−3/4		+3/4	LSB
Differential Nonlinearity	−1		+1	−1/2		+1/2	−1		+1	LSB
T_{min} to T_{max}	−1		+1	−1		+1	−1		−1	LSB
Monotonicity	GUARANTEED OVER RATED SPECIFICATION TEMPERATURE RANGE									
Unipolar Offset	−0.2		+0.2	*		*	*		*	% of FSR
Bipolar Offset	−1.0		+1.0	*		*	*		*	% of FSR
Bipolar Zero	−0.2		+0.2	*		*	*		*	% of FSR
Gain Error	−1.0		+1.0	*		*	*		*	% of FSR
TEMPERATURE COEFFICIENTS[2]										
Unipolar Offset	−5		+5	−3		+3	−5		+5	ppm of FSR/°C
Bipolar Offset	−30		+30	−20		+20	−30		+30	ppm of FSR/°C
Bipolar Zero	−15		+15	*		*	*		*	ppm of FSR/°C
Gain Drift	−50		+50	−30		+30	−50		+50	ppm of FSR/°C
Gain Drift (I_{OUT})	−150		+150	*		*	*		*	ppm of FSR/°C
DATA INPUTS										
Logic Levels (T_{min} to T_{max})										
V_{IH}	2.0		7.0	*		*	*		*	V
V_{IL}	0.0		0.8	*		*	*		*	V
Logic Currents (T_{min} to T_{max})										
I_{IH}	−10	0	+10	*	*	*	*	*	*	µA
I_{IL}	−0.5	−60	−100	*	*	*	*	−100	−200	µA
V_{TH} Pin Voltage		1.4			*			*		V
CODING	BINARY, OFFSET BINARY									
CURRENT OUTPUT RANGES	0 to 10.24, ± 5.12									mA
VOLTAGE OUTPUT RANGES	0 to 1.024, ± 0.512									V
COMPLIANCE VOLTAGE	−2		+1.2	*		*	*		*	V
OUTPUT RESISTANCE										
Exclusive of R_L	160	200	240		*			*		Ω
Inclusive of R_L	99	100	101		*			*		Ω
SETTLING TIME										
Current to										
± 0.025%		35			*			*		ns to 0.025% of FSR
± 0.1%		23			*			*		ns to 0.1% of FSR
Voltage										
50Ω Load[3], 0.512V p-p,										
to 0.025%		37			*			*		ns to 0.025% of FSR
to 0.1%		25			*			*		ns to 0.1% of FSR
to 1%		18			*			*		ns to 1% of FSR
75Ω Load[3], 0.768V p-p,										
to 0.025%		40			*			*		ns to 0.025% of FSR
to 0.1%		25			*			*		ns to 0.1% of FSR
to 1%		20			*			*		ns to 1% of FSR
100Ω (Internal R_L)[3], 1.024V p-p,										
to 0.025%		50			*			*		ns to 0.025% of FSR
to 0.1%		38			*			*		ns to 0.1% of FSR
to 1%		24			*			*		ns to 1% of FSR
Glitch Impulse[4]		350			*			*		pV-sec
Peak Amplitude		15			*			*		% of FSR
FULL-SCALE TRANSITION[5]										
10% to 90% Rise Time		11			*			*		ns
90% to 10% Fall Time		11			*			*		ns
POWER REQUIREMENTS										
+13.5V to +16.5V		27	32		*	*		*	*	mA
−13.5V to −16.5V		−7	−8		*	*		*	*	mA
Power Dissipation		525	625		*	*		*	*	mW
PSRR			0.05			*			*	% of FSR/V
TEMPERATURE RANGE										
Rated Specification[2]	0		70	0		70	−55		+125	°C
Storage	−65		+150	*		*	*		*	°C

NOTES
*Same as AD568J.
[1]Measured in I_{OUT} mode.
[2]Measured in V_{OUT} mode, unless otherwise specified. See text for further information.
[3]Total Resistance. Refer to Figure 3.
[4]At the major carry, driven by HCMOS logic. See text for further explanation.
[5]Measured in V_{OUT} mode.
Specifications shown in boldface are tested on all production units at final electrical test.
Specifications subject to change without notice.

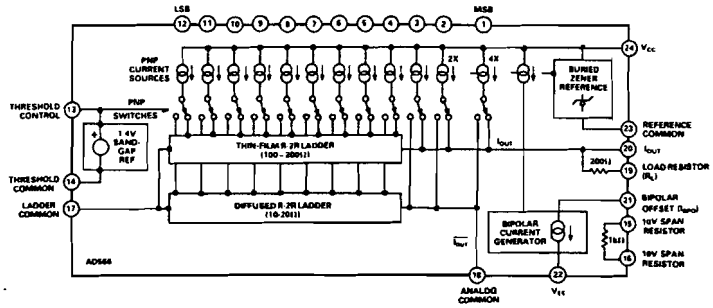
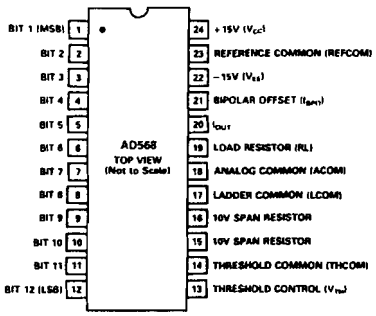


Figure 1. Functional Block Diagram

PIN CONFIGURATION



ABSOLUTE MAXIMUM RATINGS*

V_{CC} to REFCOM	0V to +18V
V_{EE} to REFCOM	0V to −18V
REFCOM to LCOM	+100mV to −10V
ACOM to LCOM	±100mV
THCOM to LCOM	±500mV
SPANs to LCOM	±12V
I_{BPO} to LCOM	±5V
I_{OUT} to LCOM	−5V to V_{TH}
Digital Inputs to THCOM	−500mV to +7.0V
Voltage Across Span Resistor	12V
V_{TH} to THCOM	−0.7V to +1.4V
Logic Threshold Control Input Current	5mA
Power Dissipation	1000mW
Storage Temperature Range	
Q (Cerdip) Package	−65°C to +150°C
Junction Temperature	175°C
Thermal Resistance	
θ_{ja}	75°C/W
θ_{jc}	25°C/W

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ORDERING GUIDE

Model[1]	Package Option[2]	Temperature Range °C	Linearity Error Max. @ 25°C	Voltage Gain T.C. Max ppm/°C
AD568JQ	24-Lead Cerdip (Q-24)	0 to +70	±1/2	±50
AD568KQ	24-Lead Cerdip (Q-24)	0 to +70	±1/4	±30
AD568SQ	24-Lead Cerdip (Q-24)	−55 to +125	±1/2	±50

NOTES
[1]For details on grade and package offerings screened in accordance with MIL-STD-883, refer to the Analog Devices Military Products Databook or current AD568/883B data sheet.
[2]Q = Cerdip. For outline information see Package Information section.

Definitions

LINEARITY ERROR (also called INTEGRAL NON-LINEARITY OR INL): Analog Devices defines linearity error as the maximum deviation of the actual analog output from the ideal output (a straight line drawn from 0 to FS) for any bit combination expressed in multiples of 1LSB. The AD568 is laser trimmed to 1/4LSB (0.006% of FS) maximum linearity error at +25°C for the K version and 1/2LSB for the J and S versions.

DIFFERENTIAL LINEARITY ERROR (also called DIFFERENTIAL NONLINEARITY or DNL): DNL is the measure of the variation in analog value, normalized to full scale, associated with a 1LSB change in digital input code. Monotonic behavior requires that the differential linearity error not exceed 1LSB in the negative direction.

MONOTONICITY: A DAC is said to be monotonic if the output either increases or remains constant as the digital input increases.

UNIPOLAR OFFSET ERROR: The deviation of the analog output from the ideal (0V or 0mA) when the inputs are set to all 0s is called unipolar offset error.

BIPOLAR OFFSET ERROR: The deviation of the analog output from the ideal (negative half-scale) when the inputs are set to all 0s is called bipolar offset error.

BIPOLAR ZERO ERROR: The deviation of the analog output from the ideal half-scale output of 0V (or 0mA) for bipolar mode when only the MSB is on (100.....00) is called bipolar zero error.

GAIN ERROR: The difference between the ideal and actual output span of FS − 1LSB, expressed in % of FS, or LSB, when all bits are on.

GLITCH IMPULSE: Asymmetrical switching times in a DAC give rise to undesired output transients which are quantified by their glitch impulse. It is specified as the net area of the glitch in nV-sec or pA-sec.

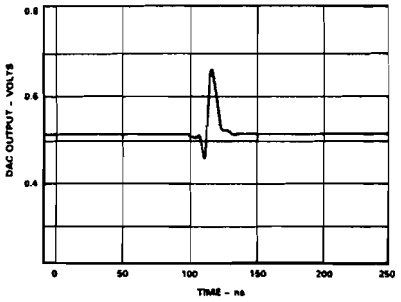


Figure 2. AD568 Glitch Impulse

COMPLIANCE VOLTAGE: The range of allowable voltage at the output of a current-output DAC which will not degrade the accuracy of the output current.

SETTLING TIME: The time required for the output to reach and remain within a specified error band about its final value, measured from the digital input transition.

Connecting the AD568

UNBUFFERED VOLTAGE OUTPUT

Unipolar Configuration

Figure 3 shows the AD568 configured to provide a unipolar 0 to +1.024V output range. In this mode, the bipolar offset terminal, Pin 21, should be grounded if not used for offset trimming.

The nominal output impedance of the AD568 with Pin 19 grounded has been trimmed to 100Ω, ± 1%. Other output impedances can be generated with an external resistor, R_{EXT}, between Pins 19 and 20. An R_{EXT} equalling 300Ω will yield a total output resistance of 75Ω, while an R_{EXT} of 100Ω will provide 50Ω of output resistance. Note that since the full-scale output current of the DAC remains 10.24mA, changing the load impedance changes the unbuffered output voltage accordingly. Settling time and full-scale range characteristics for these load impedances are provided in the specifications table.

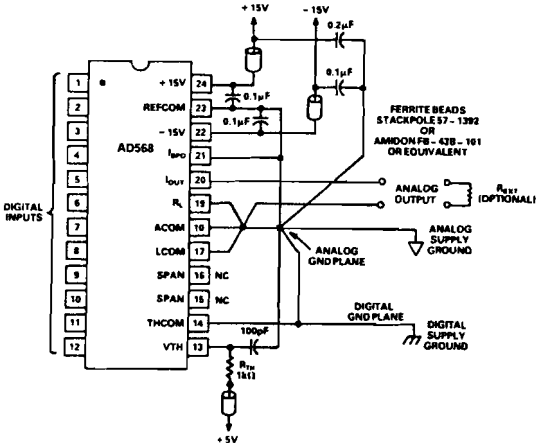


Figure 3. Unipolar Output Unbuffered 0 to +1.024V

Bipolar Configuration

Figure 4 shows the connection scheme used to provide a bipolar output voltage range of 1.024V. The bipolar offset (−0.512V) occurs when all bits are OFF (00 . . . 00), bipolar zero (0V)

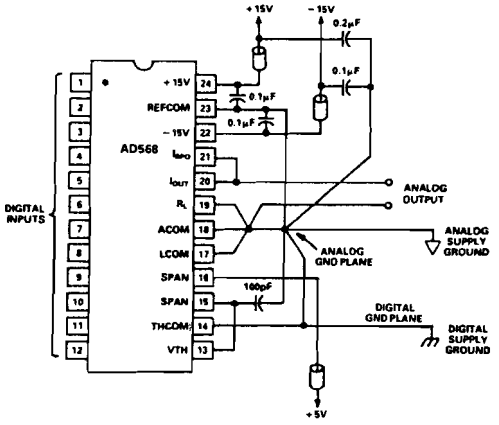


Figure 4. Bipolar Output Unbuffered ±0.512V

occurs when the MSB is ON with all other bits OFF (10 . . . 00), and full-scale minus 1LSB (0.51175V) is generated when all bits are ON (11 . . . 11). Figure 5 shows an optional bipolar mode with a 2.048V range. The scale factor in this mode will not be as accurate as the configuration shown in Figure 4, because the laser-trimmed resistor R_L is not used.

Figure 4 also demonstrates how the internal span resistor may be used to bias the V_{TH} pin (Pin 13) from a 5V supply. This eliminates the requirement for an external R_{TH} in applications that do not require the precision span resistor.

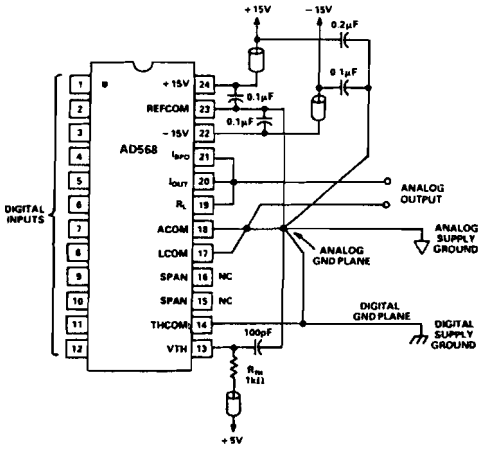


Figure 5. Bipolar Output Unbuffered ± 1.024V

Optional Gain and Zero Adjustment

The gain and offset are laser trimmed to minimize their effects on circuit performance. However, in some applications, it may be desirable to externally reduce these errors further. In those cases, the following procedures are suggested.

UNIPOLAR MODE: (Refer to Figure 6)

Step 1 – Set all bits (BIT 1–BIT 12) to Logic "0" (OFF) – note the output voltage. This is the offset error.

Step 2 – Set all bits to Logic "1" (ON). Adjust the gain trim resistor so that the output voltage is equal to the desired full scale minus 1LSB plus the offset error measured in step 1.

Step 3 – Reset all bits to Logic "0" (OFF). Adjust the offset trim resistor for 0V output.

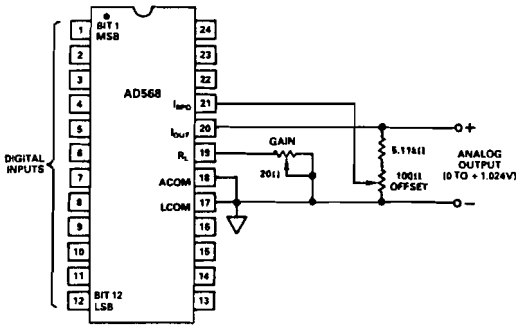


Figure 6. Unbuffered Unipolar Gain and Zero Adjust

BIPOLAR MODE (Refer to Figure 7)

Step 1 – Set bits to offset binary "zero" (10 . . . 00). Adjust the zero resistor to produce 0V at the DAC output. This removes the bipolar zero error.

Step 2 – Set all bits to Logic "1" (ON). Adjust gain trim resistor so the output voltage is equal to the desired full-scale minus 1LSB.

Step 3 – (Optional) If precise trimming of the bipolar offset is preferred to trimming of bipolar zero: set all bits to Logic "0" (OFF). Trim the zero resistor to produce the desired negative full scale at the DAC output.

Note: this may slightly compromise the bipolar zero trim.

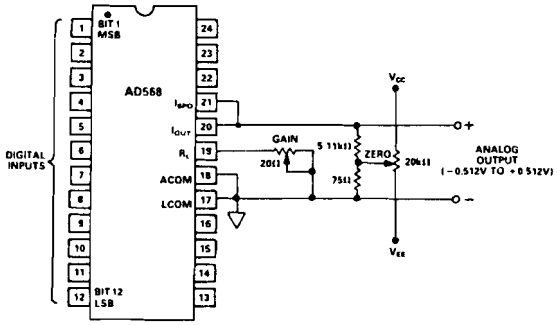


Figure 7. Bipolar Unbuffered Gain and Zero Adjust

BUFFERED VOLTAGE OUTPUT

For full-scale outputs of greater than 1V, some type of external buffer amplifier is required. The AD840 fills this requirement perfectly, settling to 0.025% from a 10V full-scale step in less than 100ns.

A 1kΩ span resistor has been provided on chip for use as a feedback resistor in buffered applications. Using R_{SPAN} (Pins 15, 16) introduces a 100mW code-dependent power source onto the chip which may generate a slight degradation in linearity. Maximum linearity performance can be realized by using an external span resistor.

Unipolar Inverting Configuration

Figure 8 shows the connections for producing a −10.24V full-scale swing. This configuration uses the AD568 in the current output mode into a summing junction at the inverting input terminal of the external op amp. With the load resistor R_L grounded, the DAC has an output impedance of 100Ω. This produces a noise gain of 11 from the noninverting terminal of the op amp, and hence, satisfies the stability criterion of the AD840 (stable at a gain of 10). The addition of a 5pF compensation capacitor across the 1kΩ feedback resistor produces optimal settling. Lower noise gain can be achieved by connecting R_L to I_{OUT}, increasing the DAC output impedance to approximately 200Ω, and reducing the noise gain to 6 (illustrated in Figure 9). While the output in this configuration will feature improved noise performance, it is somewhat less stable and may suffer from ringing. The compensation capacitance should be increased to 7pF to maintain stability at this reduced gain.

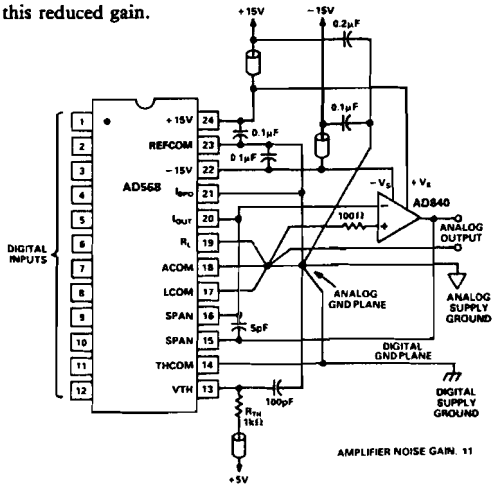


Figure 8. Unipolar Output Buffered 0 to −10.24V

Bipolar Inverting Configuration

Figure 9 illustrates the implementation of a +5.12V to −5.12V bipolar range, achieved by connecting the bipolar offset current, I_{BPO}, to the summing junction of the external amplifier. Note that since the amplifier is providing an inversion, the full-scale output voltage is −5.12V, while the bipolar offset voltage (all bits OFF) is +5.12V at the amplifier output.

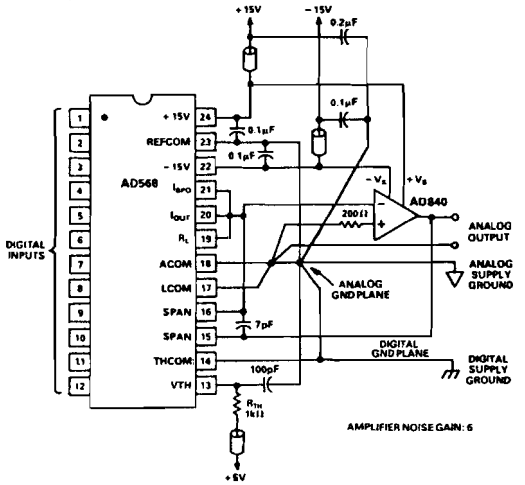


Figure 9. Bipolar Output Buffered ±5.12V

Noninverting Configuration

If a positive full-scale output voltage is required, it can be implemented using the AD568 in the unbuffered voltage output mode followed by the AD840 in a noninverting configuration (Figure 10). The noise gain of this topology is 10, requiring only 5pF across the feedback resistor to optimize settling.

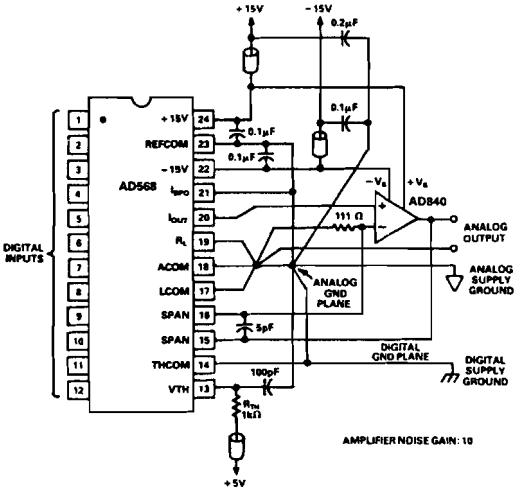


Figure 10. Unipolar Output Buffered 0 to +10.24V

Guidelines for Using the AD568

The designer who seeks to combine high speed with high precision faces a challenging design environment. Where tens of milliamperes are involved, fractions of an ohm of misplaced impedance can generate several LSBs of error. Increasing bandwidths make formerly negligible parasitic capacitances and inductances significant. As system performance reaches and exceeds that of the measurement equipment, time-honored test methods may no longer be trustworthy. The DAC's placement on the boundary between the analog and digital domains introduces additional concerns. Proper RF techniques must be used in board design, device selection, supply bypassing, grounding, and measurement if optimal performance is to be realized. The AD568 has been configured to be relatively easy to use, even in some of the more treacherous applications. The device characteristics shown in this datasheet are readily achievable if proper attention is paid to the details. Since a solid understanding of the circuit involved is one of the designer's best weapons against the difficulties of RF design, the following sections provide illustrations, explanations, examples, and suggestions to facilitate successful design with the AD568.

Current Output vs. Voltage Output

As indicated in Figures 3 through 10, the AD568 has been designed to operate in several different modes depending on the external circuit configuration. While these modes may be categorized by many different schemes, one of the most important distinctions to be made is whether the DAC is to be used to generate an output voltage or an output current. In the current output mode, the DAC output (Pin 20) is tied to some type of summing junction, and the current flowing from the DAC into this summing junction is sensed (e.g., Figures 8 and 9). In this mode, the DAC output scale is insensitive to whether the load resistor, R_L, is shorted (Pin 19 connected to Pin 20), or grounded (Pin 19 connected to Pin 18). However, this does affect the output impedance of the DAC current and may have a significant impact on the noise gain of the external circuitry. In the voltage output mode, the DAC's output current flows through its own internal impedance (perhaps in parallel with an external impedance) to generate a voltage, as in Figures 3, 4, 5, and 10. In this case, the DAC output scale is directly dependent on the load impedance. The temperature coefficient of the AD568's internal reference is trimmed in such a way that the drift of the DAC output in the voltage output mode is centered on zero. The current output of the DAC will have an additional drift factor corresponding to the absolute temperature coefficient of the internal thin-film resistors. This additional drift may be removed by judicious placement of the 1kΩ span resistor in the signal path. For example, in Figures 8 and 9, the current flowing from the DAC into the summing junction could suffer from as much as 150ppm/°C of thermal drift. However, since this current flows through the internal span resistor (Pins 15 and 16) which has a temperature coefficient that matches the DAC ladder resistors, this drift factor is compensated and the buffered voltage at the amplifier output will be within specified limits for the voltage output mode.

Output Voltage Compliance

The AD568 has a typical output compliance range of +1.2V to −2.0V (with respect to the LCOM Pin). The current-steering output stages will be unaffected by changes in the output terminal voltage over that range. However, as shown in Figure 11, there is an equivalent output impedance of 200Ω in parallel with 15pF at the output terminal which produces an equivalent error current if the voltage deviates from the ladder common. This is a linear effect which does not change with input code. Operation beyond the maximum compliance limits may cause either output stage saturation or breakdown resulting in nonlinear performance. The positive compliance limit is not affected by the positive power supply, but is a function of output current and the logic threshold voltage at V_{TH}, Pin 13.

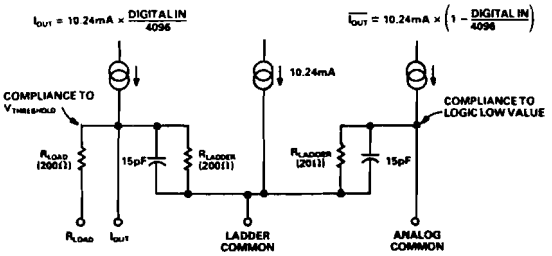


Figure 11. Equivalent Output

Digital Input Considerations

The AD568 uses a standard positive true straight binary code for unipolar outputs (all 1s full-scale output), and an offset binary code for bipolar output ranges. In the bipolar mode, with all 0s on the inputs, the output will go to negative full scale; with 111 . . . 11, the output will go to positive full scale less 1LSB; and with 100 . . 00 (only the MSB on), the output will go to zero.

The threshold of the digital inputs is set at 1.4V and does not vary with supply voltage. This is provided by a bandgap reference generator, which requires approximately 3mA of bias current achieved by tying R_{TH} to any $+V_L$ supply where

$$R_{TH} = \left(\frac{+V_L - 1.4V}{3mA}\right).$$

The input lines operate with small input currents to easily achieve interface with unbuffered CMOS logic. The digital input signals to the DAC should be isolated from the analog output as much as possible. To minimize undershoot, ringing, and possible digital feedthrough noise, the interconnect distances to the DAC inputs should be kept as short as possible. Termination resistors may improve performance if the digital lines become too long. The digital input should be free from large glitches and ringing and have maximum 10% to 90% rise and fall times of 5ns. Figure 12 shows the equivalent digital input circuit of the AD568.

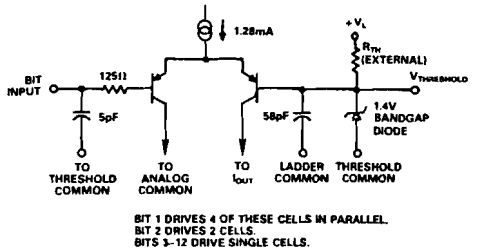


Figure 12. Equivalent Digital Input

Due to the high-speed nature of the AD568, it is recommended that high-speed logic families such as Schottky TTL, high-speed CMOS, or the new lines of FAST* TTL be used exclusively. Table I shows how DAC performance can vary depending on the driving logic used. As this table indicates, STTL, HCMOS, and FAST represent the most viable families for driving the AD568.

DAC PERFORMANCE VS. DRIVE LOGIC[1]

Logic Family	10-90% DAC Rise Time[2]	DAC SETTLING TIME[2,3] 1%	0.1%	0.025%	Glitch[4] Impulse	Maximum Glitch Excursion
TTL	11ns	18ns	34ns	50ns	2.5nV-s	240mV
LSTTL	11ns	28ns	46ns	80ns	950pV-s	160mV
STTL	9.5ns	16ns	33ns	50ns	850pV-s	150mV
HCMOS	11ns	24ns	38ns	50ns	350pV-s	115mV
FAST*	12ns	16ns	36ns	42ns	1.0nV-s	250mV

[1]All values typical, taken in test fixture diagrammed in Figure 13.
[2]Measurements are made for a 1V full-scale step into 100ΩDAC load resistance.
[3]Settling time is measured from the time the digital input crosses the threshold voltage (1.4V) to when the output is within the specified range of its final value.
[4]The worst case glitch impulse, measured on the major carry. DAC full scale is 1V.

Table I.

The variations in settling times can be attributed to differences in the rise time and current driving capabilities of the various families. Differences in the glitch impulse are predominantly dependent upon the variation in data skew. Variations in these specs occur not only between logic families, but also between different gates and latches within the same family. When selecting a gate to drive the AD568 logic input, pay particular attention to the propagation delay time specs: t_{PLH} and t_{PHL}. Selecting the smallest delays possible will help to minimize the settling time, while selection of gates where t_{PLH} and t_{PHL} are closely matched to one another will minimize the glitch impulse resulting from data skew. Of the common latches, the 74374 octal flip-flop provides the best performance in this area for many of the logic families mentioned above.

*FAST is a registered trademark of Fairchild Camera and Instrumentation Corporation.

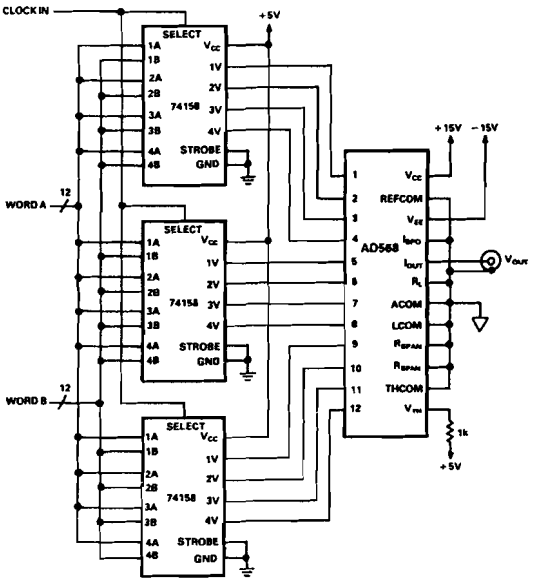


Figure 13. Test Setup for Glitch Impulse and Settling Time Measurements

Settling Time Considerations

As can be seen from Table I and the specifications page, the settling time of the AD568 is application dependent. The fastest settling is achieved in the current-output mode, since the voltage-output mode requires the output capacitance to be charged to the appropriate voltage. The DAC's relatively large output current helps to minimize this effect, but settling-time sensitive applications should avoid any unnecessary parasitic capacitance at the output node of voltage output configurations. Direct measurement of the fine scale DAC settling time, even in the voltage output mode, is extremely tricky: analog scope front ends are generally incapable of recovering from overdrive quickly enough to give an accurate settling representation. The plot shown in Figure 14 was obtained using Data Precision's 640 16-bit sampling head, which features the quick overdrive recovery characteristic of sampling approaches combined with high accuracy and relatively small thermal tail.

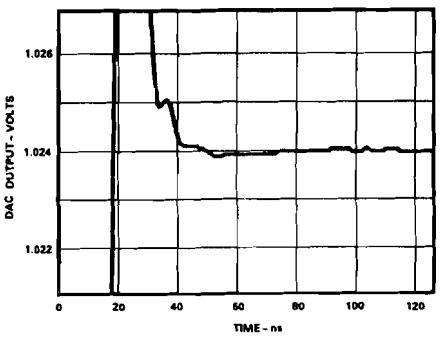


Figure 14. Zero to Full-Scale Settling

Glitch Considerations

In many high-speed DAC applications, glitch performance is a critical specification. In a conventional DAC architecture such as the AD568 there are two basic glitch mechanisms: data skew and digital feedthrough. A thorough understanding of these sources can help the user to minimize glitch in any application.

DIGITAL FEEDTHROUGH – As with any converter product, a high-speed digital-to-analog converter is forced to exist on the frontier between the noisy environment of high-speed digital logic and the sensitive analog domain. The problems of this interfacing are particularly acute when demands of high speed (greater than 10MHz switching times) and high precision (12 bits or more) are combined. No amount of design effort can perfectly isolate the analog portions of a DAC from the spectral components of a digital input signal with a 2ns risetime. Inevitably, once this digital signal is brought onto the chip, some of its higher frequency components will find their way to the sensitive analog nodes, producing a digital feedthrough glitch. To minimize the exposure to this effect, the AD568 has intentionally omitted the on-board latches that have been included in many slower DACs. This not only reduces the overall level of digital activity on chip, it also avoids bringing a latch clock pulse on board, whose opposite edge inevitably produces a substantial glitch, even when the DAC is not supposed to be changing codes. Another path for digital noise to find its way onto a converter chip is through the reference input pin. The completely internal reference featured in the AD568 eliminates this noise input, providing a greater degree of signal integrity in the analog portions of the chip.

DATA SKEW – The AD568, like many of its slower predecessors, essentially uses each digital input line to switch a separate, weighted current to either the output (I_{OUT}) or some other node (ANALOG COM). If the input bits are not changed simultaneously, or if the different DAC bits switch at different speeds, then the DAC output current will momentarily take on some incorrect value. This effect is particularly troublesome at the "carry points", where the DAC output is to change by only one LSB, but several of the larger current sources must be switched to realize this change. Data skew can allow the DAC output to move a substantial amount towards full scale or zero (depending upon the direction of the skew) when only a small transition is desired. Great care was taken in the design and layout of the AD568 to ensure that switching times of the DAC switches are symmetrical and that the length of the input data lines are short and well matched. The glitch-sensitive user should be equally diligent about minimizing the data skew at the AD568's inputs, particularly for the 4 or 5 most significant bits. This can be achieved by using the proper logic family and gate to drive the DAC, and keeping the interconnect lines between the logic outputs and the DAC inputs as short and as well matched as possible, particularly for the most significant bits. The top 6 bits should be driven from the same latch chip if latches are used.

Glitch Reduction Schemes

BIT-DESKEWING – Even carefully laid-out boards using the proper driving logic may suffer from some degree of data-skew induced glitch. One common approach to reducing this effect is to add some appropriate capacitance (usually several pF) to each of the 2 or 3 most significant bits. The exact value of each capacitor for a given application should be determined experimentally, as it will be dependent on circuit board layout and the type of driving logic used. Table II presents a few examples of how the glitch impulse may be reduced through passive deskewing.

BIT DELAY GLITCH REDUCTION EXAMPLES[1]

Logic Family	Gate	Uncompensated Glitch	Compensation Used	Compensated Glitch
HCMOS	74157	350pV-s	C2 = 5pF	250pV-s
STTL	74158	850pV-s	R1 = 50Ω, C1 = 7pF	600pV-s

NOTE
[1]Measurements were made using a modified version of the fixture shown in Figure 13, with resistors and capacitors placed as shown in Figure 15. Resistance and capacitance values were set to zero except as noted.

Table II.

As Figure 15 indicates, in some cases it may prove useful to place a few hundred ohms of series resistance in the input line to enhance the delay effect. This approach also helps to reduce some of the digital feedthrough glitch, as the higher frequency spectral components are being filtered out of the most significant bits' digital inputs.

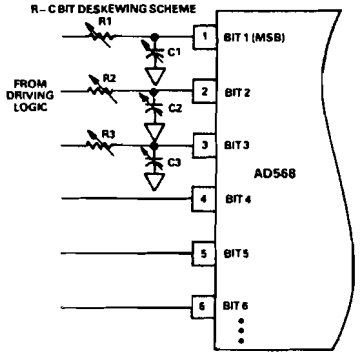


Figure 15. R–C Bit Deskewing Scheme

THRESHOLD SHIFT – It is also possible to reduce the data skew by shifting the level of logic voltage threshold, V_{TH} (Pin 13). This can be readily accomplished by inserting some resistance between the THRESHOLD COM pin (Pin 14) and ground, as in Figure 16. To generate threshold voltages below 1.4V, Pin 13 may be directly driven with a voltage source, leaving Pin 14 tied to the ground plane. As Note 2 in Table III indicates, lowering the threshold voltage may reduce output voltage compliance below the specified limits, which may be of concern in an unbuffered voltage output topology.

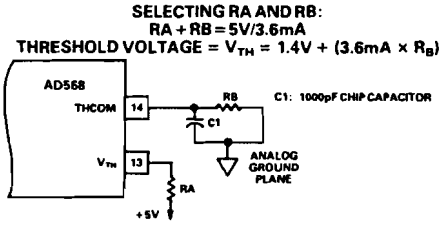


Figure 16. Positive Threshold Voltage Shift

Table III shows the glitch reduction achieved by shifting the threshold voltage for HCMOS, STTL, and FAST logic.

THRESHOLD SHIFT FOR GLITCH IMPROVEMENT[1]

Logic Family	Gate	Uncompensated Glitch	Modified Threshold[2]	Resulting Glitch
HCMOS	74HC158	350pV-s	1.7V	150pV-s
STTL	74S158	850pV-s	1.0V	200pV-s
FAST	74F158	1000pV-s	1.3V	480pV-s

NOTES
[1]Measurements made on a modified version of the circuit shown in Figure 13, with a 1V full scale.
[2]Use care in any scheme that lowers the threshold voltage since the output voltage compliance of the DAC is sensitive to this voltage. If the DAC is to be operated in the voltage output mode, it is strongly suggested that the threshold voltage be set at least 200mV above the output voltage full scale.

Table III.

Deglitching

Some applications may prove so sensitive to glitch impulse that reduction of glitch impulse by an order of magnitude or more is required. In order to realize glitch impulses this low, some sort of sample-and-hold amplifier (SHA)-based deglitching scheme must be used.

There are high-speed SHAs available with specifications sufficient to deglitch the AD568, however most are hybrid in design at costs which can be prohibitive. A high performance, low cost alternative shown in Figure 17 is a discrete SHA utilizing a high-speed monolithic op amp and high-speed DMOS FET switches.

This SHA circuit uses the inverting integrator architecture. The AD841 operational amplifier used (300MHz gain bandwidth product) is fabricated on the same high-speed process as the AD568. The time constant formed by the 200Ω resistor and the 100pF capacitor determines the acquisition time and also band limits the output signal to eliminate slew induced distortion.

A discrete drive circuit is used to achieve the best performance from the SD5000 quad DMOS switch. This switch driving cell is composed of MPS571 RF npn transistors and an MC10124 TTL to ECL translator. Using this technique provides both high speed and highly symmetrical drive signals for the SD5000 switches. The switches are arranged in a single-throw double-pole (SPDT) configuration. The 360pF "flyback" capacitor is switched to the op amp summing junction during the hold mode to keep switching transients from feeding to the output. This capacitor is grounded during sample mode to minimize its effect on acquisition time.

Circuit layout for a high speed SHA is almost as critical as the design itself. Figure 17 shows a recommended layout of the deglitching cell for a double sided printed circuit board. The layout is very compact with care taken that all critical signal paths are short.

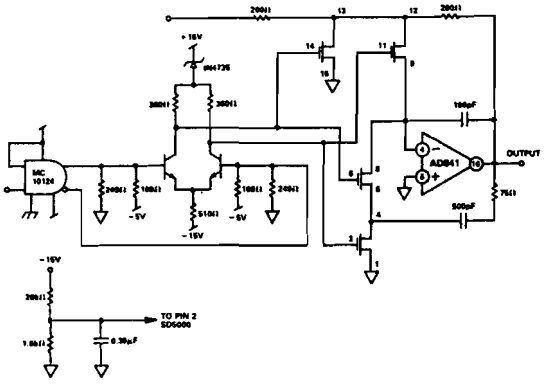


Figure 17. High Performance Deglitcher

Grounding Rules

The AD568 brings out separate reference, output, and digital power grounds. This allows for optimum management of signal ground currents for low noise and high-speed settling performance. The separate ground returns are provided to minimize changes in current flow in the analog signal paths. In this way, logic return currents are not summed into the same return path with the analog signals.

It is important to understand which supply and signal currents are flowing in which grounds so that they may be returned to the proper power supply in the best possible way.

The majority of the current that flows into the V_{CC} supply (Pin 24) flows out (depending on the DAC input code) either the ANALOG COMMON (Pin 18), the LADDER COMMON (Pin 17), and/or I_{OUT} (Pin 20).

The current in the LADDER COMMON is configured to be code independent when the output current is being summed into a virtual ground. If I_{OUT} is operated into its own output impedance (or in any unbuffered voltage output mode) the current in LADDER COMMON will become partially code dependent.

The current in the ANALOG COMMON (Pin 18) is an approximate complement of the current in I_{OUT}, i.e., zero when the DAC is at full scale and approximately 10mA at zero input code.

A relatively constant current (not code dependent) flows out the REFERENCE COMMON (Pin 23).

The current flowing out of the V_{EE} supply (Pin 22) comes from a combination of reference ground and BIPOLAR OFFSET (Pin 21). The plus and minus 15V supplies are decoupled to the REFERENCE COMMON.

The ground side of the load resistor R_L, ANALOG COMMON and LADDER COMMON should be tied together as close to the package pins as possible. The analog output voltage is then referred to this node and thus it becomes the "high quality" ground for the AD568. The REFERENCE COMMON (and Bipolar offset when not used), should also be connected to this node.

All of the current that flows into the V_{TH} terminal (Pin 13) from the resistor tied to the 5V logic supply (or other convenient positive supply) flows out the THRESHOLD COMMON (Pin 14). This ground pin should be returned directly to the digital ground plane on its own individual line.

The +5V logic supply should be decoupled to the THRESHOLD COMMON.

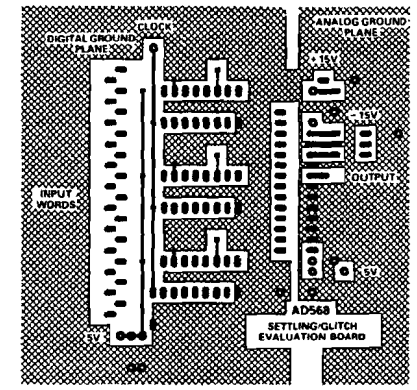
Because the V_{TH} pin is connected directly to the DAC switches it should be decoupled to the analog output signal common.

In order to preserve proper operation of the DAC switches, the digital and analog grounds need to eventually be tied together. This connection between the ground planes should be made within 1/2" of the DAC.

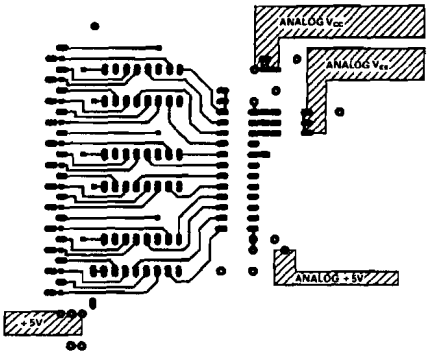
The Use of Ground and Power Planes

If used properly, ground planes can perform a myriad of functions on high-speed circuit boards: bypassing, shielding, current transport, etc. In mixed signal design, the analog and digital portions of the board should be distinct from one another, with the analog ground plane covering analog signal traces and the digital ground plane confined to areas covering digital interconnect.

The two ground planes should be connected at or near the DAC. Care should be taken to insure that the ground plane is uninterrupted over crucial signal paths. On the digital side, this includes the digital input lines running to the DAC and any clock lines. On the analog side, this includes the DAC output signal as well as the supply feeders. The use of wide runs or planes in the routing of power lines is also recommended. This serves the dual function of providing a low series impedance power supply to the part as well as providing some "free" capacitive decoupling to the appropriate ground plane. Figure 18 illustrates the PC board used for the circuit shown in Figure 13. This design was constructed on a simple two-layer board and illustrates many of the points discussed above. If more layers of interconnect are available, even better results are possible.



Component Side



Foil Side

Figure 18. Printed Circuit Board Layout

Using The Right Bypass Capacitors

Probably the most important external components associated with any high-speed design are the capacitors used to bypass the power supplies. Both selection and placement of these capacitors can be critical and, to a large extent, dependent upon the specifics of the system configurations. The dominant consideration in selection of bypass capacitors for the AD568 is minimization of series resistance and inductance. Many capacitors will begin to look inductive at 20MHz and above, the very frequencies we are most interested in bypassing. Ceramic and film-type capacitors generally feature lower series inductance than tantalum or electrolytic types. A few general rules are of universal use when approaching the problem of bypassing:

Bypass capacitors should be installed on the printed circuit board with the shortest possible leads consistent with reliable construction. This helps to minimize series inductance in the leads. Chip capacitors are optimal in this respect.

Some series inductance between the DAC supply pins and the power supply plane often helps to filter out high-frequency power supply noise. This inductance can be generated using a small ferrite bead.

High-Speed Interconnect and Routing

It is essential that care be taken in the signal and power ground circuits to avoid inducing extraneous voltage drops in the signal ground paths. It is suggested that all connections be short and direct, and as physically close to the package as possible, so that the length of any conduction path shared by external components will be minimized. When runs exceed an inch or so in length, some type of termination resistor may be required. The necessity and value of this resistor will be dependent upon the logic family used.

For maximum ac performance, the DAC should be mounted directly to the circuit board; sockets should not be used as they introduce unwanted capacitive coupling between adjacent pins of the device.

Applications

1μs, 12-BIT SUCCESSIVE APPROXIMATION A/D CONVERTER

The AD568's unique combination of high speed and true 12-bit accuracy can be used to construct a 12-bit SAR-type A/D converter with a sub-μs conversion time. Figure 19 shows the configuration used for this application. A negative analog input voltage is converted into current and brought into a summing junction with the DAC current. This summing junction is bidirectionally clamped with two Shottky diodes to limit its voltage excursion from ground. This voltage is differentially amplified and passed to a high-speed comparator. The comparator output is latched and fed back to the successive approximation register, which is then clocked to generate the next set of codes for the DAC.

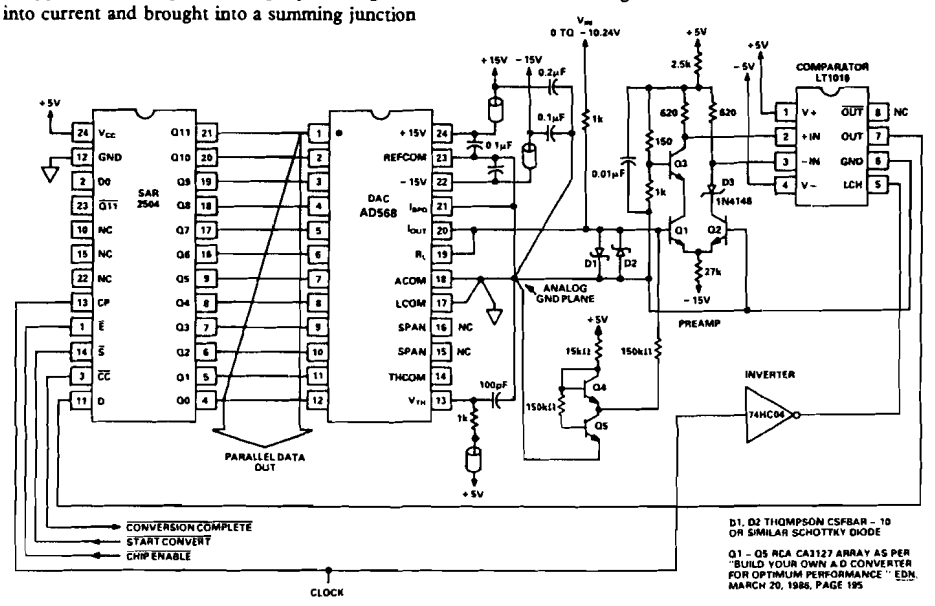


Figure 19. AD568 1μs Successive Approximation A/D Application

Circuit Details

Figure 20 shows an approximate timing budget for the A/D converter. If 12 cycles are to be completed in 1μs, approximately 80ns is allowed for each cycle. Since the Shottky diodes clamp the voltage of the summing junction, the DAC settling time approaches the current-settling value of 35ns, and hence uses up less than half the timing budget.

To maintain simplicity, a simple clock is used that runs at a constant rate throughout the conversion, with a duty cycle of approximately 90%. If absolute speed is worth the additional complexity, the clock frequency can be increased as the conversion progresses since the DAC must settle from increasingly smaller steps.

When seeking a cycle time of less than 100ns, the delays generated by the older generation SAR registers become problematic. Newer, higher speed SAR logic chips are becoming available in the classic 2504 pinout that cuts the logic overhead in half. One example of this is Zyrel's ZR2504.

Finding a comparator capable of keeping up with this DAC arrangement is fairly difficult: it must respond to an overdrive of 250μV (1LSB) in less than 25ns. Since no inexpensive comparator exists with these specs, special arrangements must be made. The LT1016 comparator provides relatively quick response, but requires at least 5mV of overdrive to maintain this speed. A discrete preamplifier may be used to amplify the summing junction voltage to sufficiently overdrive the comparator. Care must be exercised in the layout of the preamp/comparator block to avoid introducing comparator instability with the preamp's additional gain.

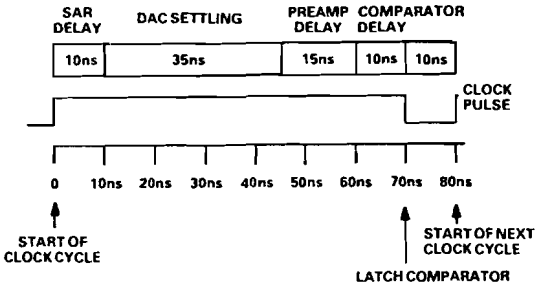


Figure 20. Typical Clock Cycle for a 1μs SAR A/D Converter

HIGH-SPEED MULTIPLYING DAC

A powerful use for the AD568 is found in multiplying applications, where the DAC controls the amplitude of a high-speed signal. Specifically, using the AD568 as the control voltage input signal for the AD539 60MHz analog multiplier and AD5539 wide-band op amp, a high-speed multiplying DAC can be built.

In the application shown in Figure 21, the AD568 is used in a buffered voltage output mode to generate the input to the AD539's control channel. The speed of the AD568 allows oversampling of the control signal waveform voltage, thereby providing increased spectral purity of the amplitude envelope that modulates the analog input channels.

The AD568 is configured in the unbuffered unipolar output mode. The internal 200Ω load resistor creates the 0-1V FS output signal, which is buffered and amplified to a 0-3V range suitable for the control channel of the AD539.

A 500Ω input impedance exists at Pin 1, the input channel. To provide a buffer for the 0-1V output signal from the AD568 looking into the impedance and to achieve the full-scale range, the AD841, high-speed, fast settling op amp is included. The gain of 3 is achieved with a 2kΩ resistor configured in follower mode with a 1kΩ pot and 500Ω resistor. A 20kΩ pot with connections to Pins 3, 4 and 12 is provided for offset trim.

The AD539 can accept two separate input signals, each with a nominal full-scale voltage range of ±2V. Each signal can then be simultaneously controlled by the AD568 signal at the common input channel, V_X. The current outputs from the two signal channels, Pins 11 and 14, applied to the AD5539 in a subtracting configuration, provide the voltage output signal:

$$V_{OUT} = \frac{D}{4096} \times \frac{V_{Y1} - V_{Y2}}{2V} \quad (0 \leq D \leq 4095)$$

For applications where only a single channel is involved, channel 2, V_{Y2}, is tied to ground. This provides:

$$V_{OUT} = \frac{D}{4096} \times \frac{V_{Y1}}{2V} \quad (0 \leq D \leq 4095)$$

Some AD539 circuit details: The control amplifier compensation capacitor for Pin 2, C_C, must have a minimum value of 3000pF to provide circuit stability. For improved bandwidth and feedthrough, the feedthrough capacitor between Pins 1 and 2 should be 5-20% of C_C. A Schottky diode at Pin 2 can improve recovery time from small negative values of V_X. Lead lengths along the path of the high-speed signal from AD568 should be kept at a minimum.

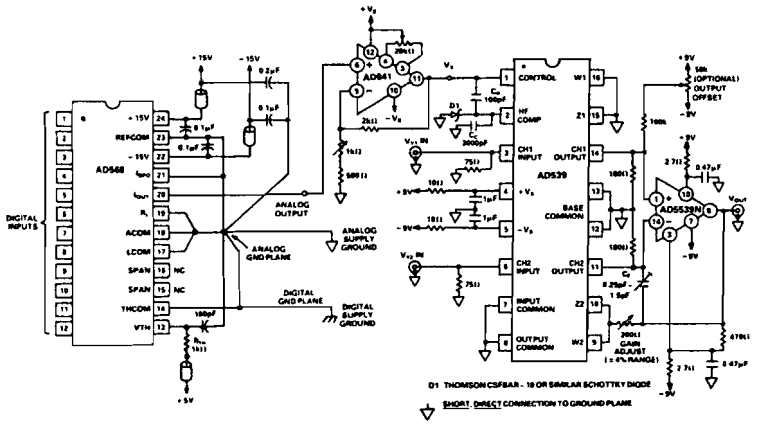


Figure 21. Wideband Digitally Controlled Multiplier

ANALOG DEVICES

Wideband, Fast Settling Op Amp

AD840

FEATURES

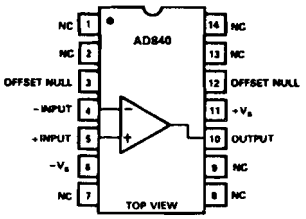
Wideband AC Performance
- **Gain Bandwidth Product: 400 MHz (Gain ≥ 10)**
- **Fast Settling: 100 ns to 0.01% for a 10 V Step**
- **Slew Rate: 400 V/μs**
- **Stable at Gains of 10 or Greater**
- **Full Power Bandwidth: 6.4 MHz for 20 V p–p into a 500 Ω Load**

Precision DC Performance
- **Input Offset Voltage: 0.3 mV max**
- **Input Offset Drift: 3 μV/°C typ**
- **Input Voltage Noise: 4 nV/$\sqrt{Hz}$**
- **Open-Loop Gain: 130 V/mV into a 1 kΩ Load**
- **Output Current: 50 mA min**
- **Supply Current: 12 mA max**

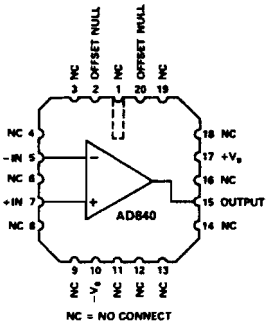
APPLICATIONS
- **Video and Pulse Amplifiers**
- **DAC and ADC Buffers**
- **Line Drivers**
- **Available in 14-Pin Plastic DIP, Hermetic Cerdip and 20-Pin LCC Packages and in Chip Form**
- **MIL-STD-883B Processing Available**

CONNECTION DIAGRAMS

Plastic DIP (N) Package and Cerdip (Q) Package



LCC (E) Package



PRODUCT DESCRIPTION

The AD840 is a member of the Analog Devices' family of wide bandwidth operational amplifiers. This high speed/high precision family includes, among others, the AD841, which is unity-gain stable, and the AD842, which is stable at a gain of two or greater and has 100 mA minimum output current drive. These devices are fabricated using Analog Devices' junction isolated complementary bipolar (CB) process. This process permits a combination of dc precision and wideband ac performance previously unobtainable in a monolithic op amp. In addition to its 400 MHz gain bandwidth product, the AD840 offers extremely fast settling characteristics, typically settling to within 0.01% of final value in 100 ns for a 10 volt step.

The AD840 remains stable over its full operating temperature range at closed-loop gains of 10 or greater. It also offers a low quiescent current of 12 mA maximum, a minimum output current drive capability of 50 mA, a low input voltage noise of 4 nV/$\sqrt{Hz}$ and a low input offset voltage of 0.3 mV maximum (AD840K).

The 400 V/μs slew rate of the AD840, along with its 400 MHz gain bandwidth, ensures excellent performance in video and pulse amplifier applications. This amplifier is ideally suited for use in high frequency signal conditioning circuits and wide bandwidth active filters. The extremely rapid settling time of the AD840 makes it the preferred choice for data acquisition applications which require 12-bit accuracy. The AD840 is also appropriate for other applications such as high speed DAC and ADC buffer amplifiers and other wide bandwidth circuitry.

APPLICATION HIGHLIGHTS

1. The high slew rate and fast settling time of the AD840 make it ideal for DAC and ADC buffers, line drivers and all types of video instrumentation circuitry.
2. The AD840 is truly a precision amplifier. It offers 12-bit accuracy to 0.01% or better and wide bandwidth, performance previously available only in hybrids.
3. The AD840's thermally balanced layout and the high speed of the CB process allow the AD840 to settle to 0.01% in 100 ns without the long "tails" that occur with other fast op amps.
4. Laser wafer trimming reduces the input offset voltage to 0.3 mV max on the K grade, thus eliminating the need for external offset nulling in many applications. Offset null pins are provided for additional versatility.
5. Full differential inputs provide outstanding performance in all standard high frequency op amp applications where circuit gain will be 10 or greater.
6. The AD840 is an enhanced replacement for the HA2540.

AD840—SPECIFICATIONS (@ +25°C and ±15 V dc, unless otherwise noted)

Model	Conditions	AD840J Min	AD840J Typ	AD840J Max	AD840K Min	AD840K Typ	AD840K Max	AD840S Min	AD840S Typ	AD840S Max	Units
INPUT OFFSET VOLTAGE[1]			0.2	1		0.1	0.3		0.2	1	mV
	T_{min} - T_{max}			1.5			0.7			2	mV
Offset Drift			5			3			5		μV/°C
INPUT BIAS CURRENT			3.5	8		3.5	5		3.5	8	μA
	T_{min} - T_{max}			10			6			12	μA
INPUT OFFSET CURRENT			0.1	0.4		0.1	0.2		0.1	0.4	μA
	T_{min} - T_{max}			0.5			0.3			0.6	μA
INPUT CHARACTERISTICS	Differential Mode										
Input Resistance			30			30			30		kΩ
Input Capacitance			2			2			2		pF
INPUT VOLTAGE RANGE											
Common Mode		±10	12		±10	12		±10	12		V
Common-Mode Rejection	V_{CM} = ±10 V	90	110		106	115		90	110		dB
	T_{min} - T_{max}	85			90			85			dB
INPUT VOLTAGE NOISE	f = 1 kHz		4			4			4		nV/√Hz
Wideband Noise	10 Hz to 10 MHz		10			10			10		μV rms
OPEN LOOP GAIN	V_O = ±10 V										
	R_{LOAD} = 1 kΩ	100	130		100	130		100	130		V/mV
	T_{min} - T_{max}	50	80		75	100		50	80		V/mV
	R_{LOAD} = 500 Ω	75			100			75			V/mV
	T_{min} - T_{max}	50			75			50			V/mV
OUTPUT CHARACTERISTICS											
Voltage	R_{LOAD} ≥ 500 Ω										
	T_{min} - T_{max}	±10			±10			±10			V
Current	V_{OUT} = ±10 V	50			50			50			mA
Output Resistance	Open Loop		15			15			15		Ω
FREQUENCY RESPONSE											
Gain Bandwidth Product	V_{OUT} = 90 mV p-p										
	A_V = −10		400			400			400		MHz
Full Power Bandwidth[2]	V_O = 20 V p-p										
	R_{LOAD} ≥ 500 Ω	5.5	6.4		5.5	6.4		5.5	6.4		MHz
Rise Time	A_V = −10		10			10			10		ns
Overshoot[3]	A_V = −10		20			20			20		%
Slew Rate[3]	A_V = −10	350	400		350	400		350	400		V/μs
Settling Time[3] −10 V Step	A_V = −10										
	to 0.1%		80			80			80		ns
	to 0.01%		100			100			100		ns
OVERDRIVE RECOVERY	−Overdrive		190			190			190		ns
	+Overdrive		350			350			350		ns
DIFFERENTIAL GAIN	f = 4.4 MHz		0.025			0.025			0.025		%
DIFFERENTIAL PHASE	f = 4.4 MHz		0.04			0.04			0.04		Degree
POWER SUPPLY											
Rated Performance			±15			±15			±15		V
Operating Range		±5		±18	±5		±18	±5		±18	V
Quiescent Current			12	14		12	14		12	14	mA
	T_{min} - T_{max}			16			16			18	mA
Power Supply Rejection Ratio	V_S = ±5 V to ±18 V	90	100		94	100		90	100		dB
	T_{min} - T_{max}	80			86			80			dB
TEMPERATURE RANGE											
Rated Performance[4]		0		+75	0		+75	−55		+125	°C
TRANSISTOR COUNT	# of Transistors		72			72			72		

NOTES
[1]Input offset voltage specifications are guaranteed after 5 minutes at T_A = +25°C.
[2]Full power bandwidth = slew rate/2π V_{PEAK}.
[3]Refer to Figures 22 and 23.
[4]"S" grade T_{min}-T_{max} specifications are tested with automatic test equipment at T_A = −55°C and T_A = +125°C.
All min and max specifications are guaranteed. Specifications shown in **boldface** are tested on all production units.
Specifications subject to change without notice.

ABSOLUTE MAXIMUM RATINGS[1]

Supply Voltage . . . . . ±18 V
Internal Power Dissipation[2]
Plastic (N) . . . . . 1.5 W
Cerdip (Q) . . . . . 1.3 W
LCC (E) . . . . . 1.0 W
Input Voltage . . . . . $\pm V_S$
Differential Input Voltage . . . . . ±6 V
Storage Temperature Range
Q, E . . . . . −65°C to +150°C
N . . . . . −65°C to +125°C
Junction Temperature (T_J) . . . . . +175°C
Lead Temperature Range (Soldering 60 sec) . . . . . +300°C

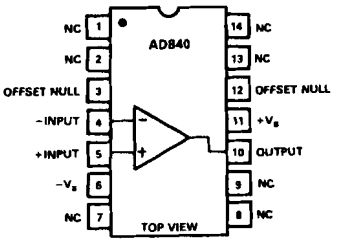
NOTES
[1]Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.
[2]Maximum internal power dissipation is specified so that T_J does not exceed +175°C at an ambient temperature of +25°C.

Thermal Characteristics:

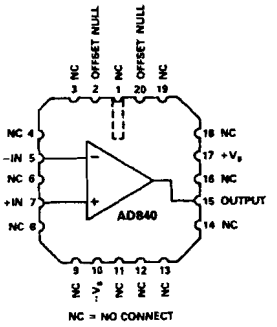
	θ_{JC}	θ_{JA}	Derate at
Cerdip Package	30°C/W	110°C/W	8.7 mW/°C
Plastic Package	30°C/W	100°C/W	10 mW/°C
LCC Package	35°C/W	150°C/W	6.7 mW/°C

Recommended Heat Sink:
Aavid Engineering© #602B

Plastic DIP (N) Package and Cerdip (Q) Package



LCC (E) Package



AD840 Connection Diagrams

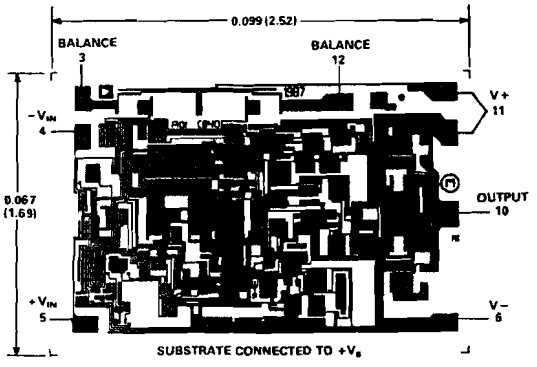
ORDERING GUIDE

Model[1]	Package Options[2]
AD840JN	N-14
AD840KN	N-14
AD840JQ	Q-14
AD840KQ	Q-14
AD840SQ	Q-14
AD840SQ-883B	Q-14
AD840SE-883B	E-20A

NOTES
[1]J and S Grade Chips also available.
[2]N = Plastic DIP; Q = Cerdip; E = LCC (Leadless Ceramic Chip Carrier). For outline information see Package Information section.

METALIZATION PHOTOGRAPH

Contact factory for latest dimensions.
Dimensions shown in inches and (mm).



AD840—Typical Characteristics (at +25°C and V_S = ±15 V, unless otherwise noted)

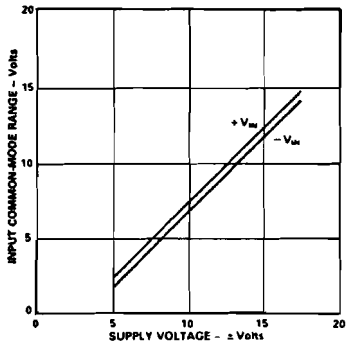


Figure 1. Input Common-Mode Range vs. Supply Voltage

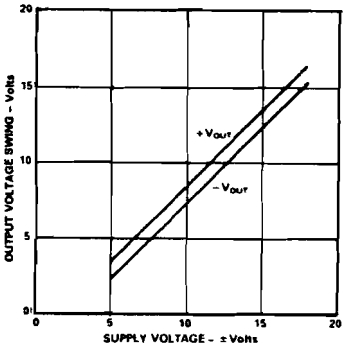


Figure 2. Output Voltage Swing vs. Supply Voltage

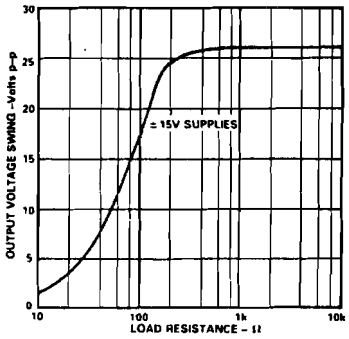


Figure 3. Output Voltage Swing vs. Load Resistance

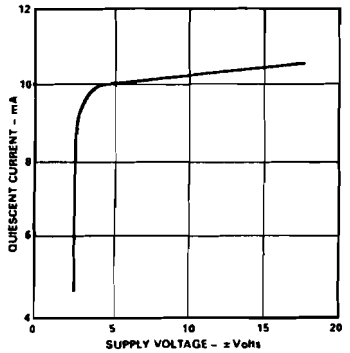


Figure 4. Quiescent Current vs. Supply Voltage

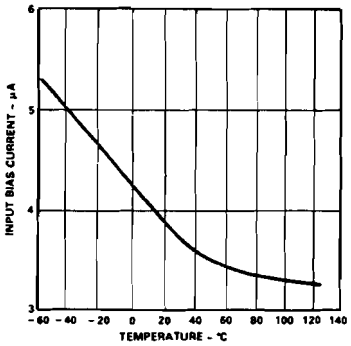


Figure 5. Input Bias Current vs. Temperature

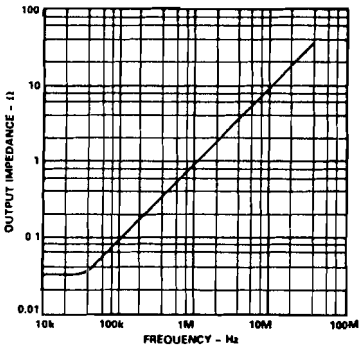


Figure 6. Output Impedance vs. Frequency

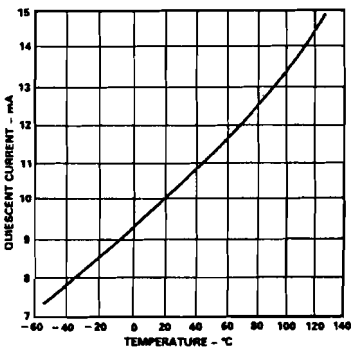


Figure 7. Quiescent Current vs. Temperature

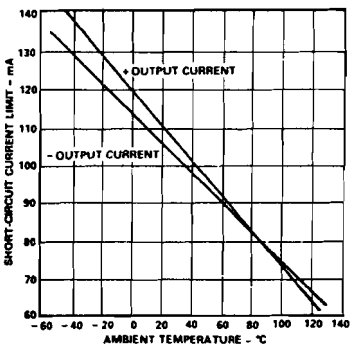


Figure 8. Short-Circuit Current Limit vs. Temperature

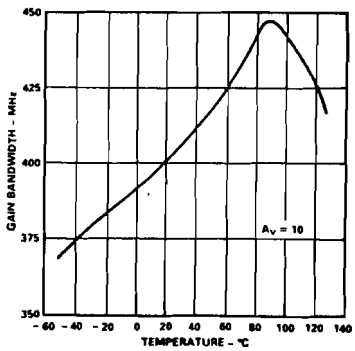


Figure 9. Gain Bandwidth Product vs. Temperature

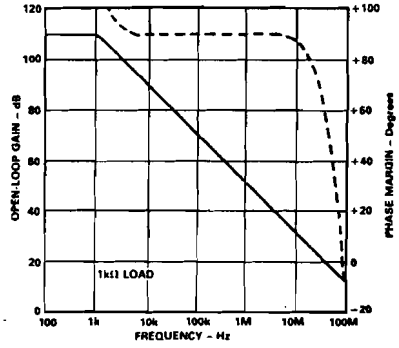


Figure 10. Open-Loop Gain and Phase Margin Phase vs. Frequency

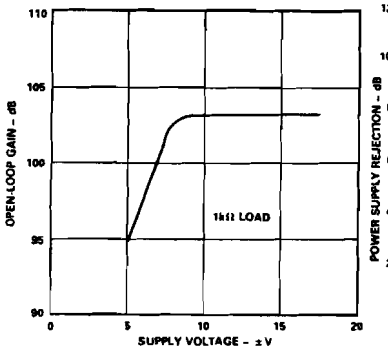


Figure 11. Open-Loop Gain vs. Supply Voltage

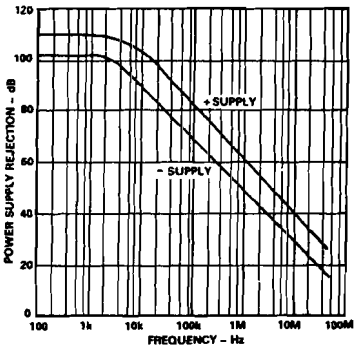


Figure 12. Power Supply Rejection vs. Frequency

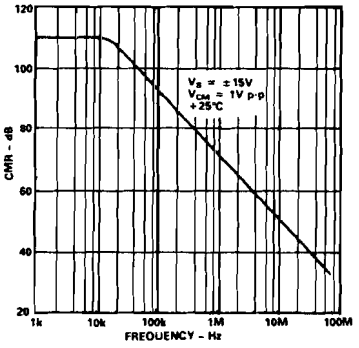


Figure 13. Common-Mode Rejection vs. Frequency

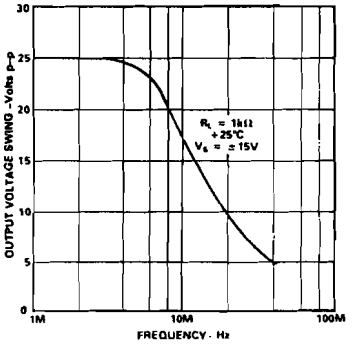


Figure 14. Large Signal Frequency Response

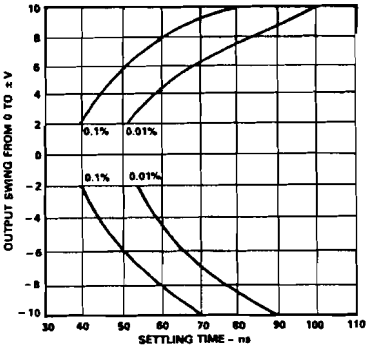


Figure 15. Output Swing and Error vs. Settling Time

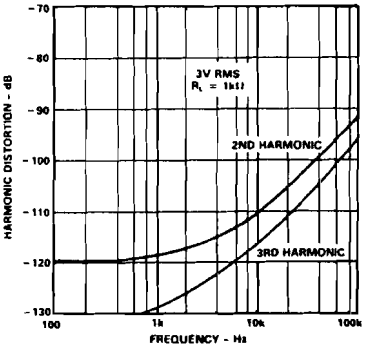


Figure 16. Harmonic Distortion vs. Frequency

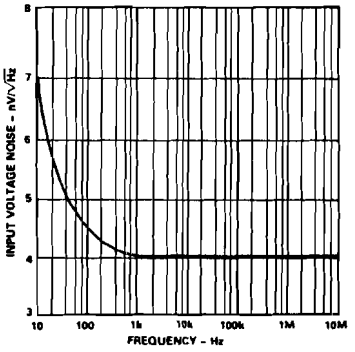


Figure 17. Input Voltage Noise Spectral Density

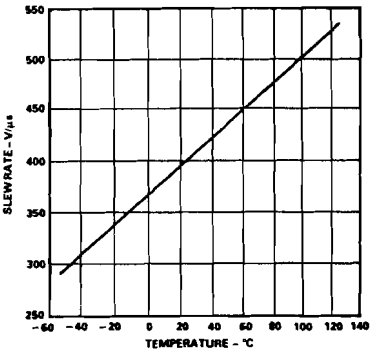


Figure 18. Slew Rate vs. Temperature

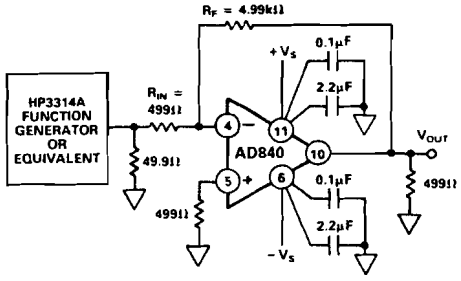


Figure 19a. Inverting Amplifier Configuration (DIP Pinout)

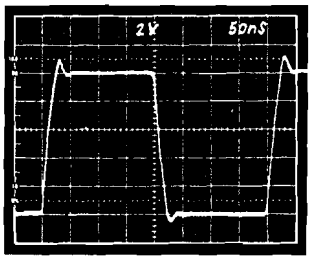


Figure 19b. Inverter Large Signal Pulse Response

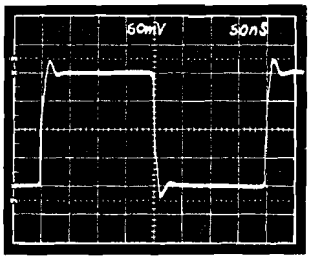


Figure 19c. Inverter Small Signal Pulse Response

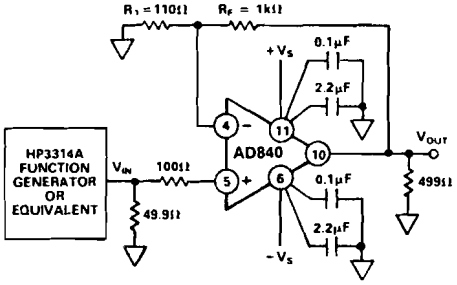


Figure 20a. Noninverting Amplifier Configuration (DIP Pinout)

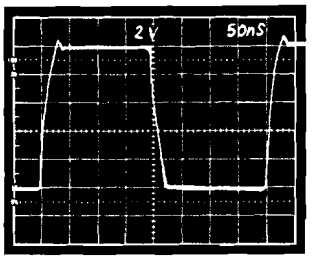


Figure 20b. Noninverting Large Signal Pulse Response

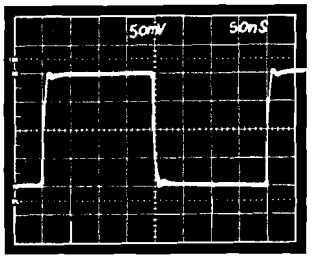


Figure 20c. Noninverting Small Signal Pulse Response

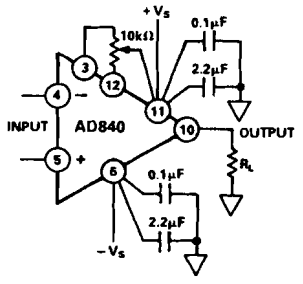


Figure 21. Offset Nulling (DIP Pinout)

OFFSET NULLING

The input offset voltage of the AD840 is very low for a high speed op amp, but if additional nulling is required, the circuit shown in Figure 21 can be used.

AD840 SETTLING TIME

Figures 22 and 24 show the settling performance of the AD840 in the test circuit shown in Figure 23.

Settling time is defined as:

> The interval of time from the application of an ideal step function input until the closed-loop amplifier output has entered and remains within a specified error band.

This definition encompasses the major components which comprise settling time. They include (1) propagation delay through the amplifier; (2) slewing time to approach the final output value; (3) the time of recovery from the overload associated with slewing; and (4) linear settling to within the specified error band.

Expressed in these terms, the measurement of settling time is obviously a challenge and needs to be done accurately to assure the user that the amplifier is worth consideration for the application.

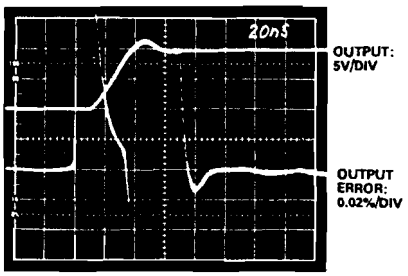


Figure 22. AD840 0.01% Settling Time

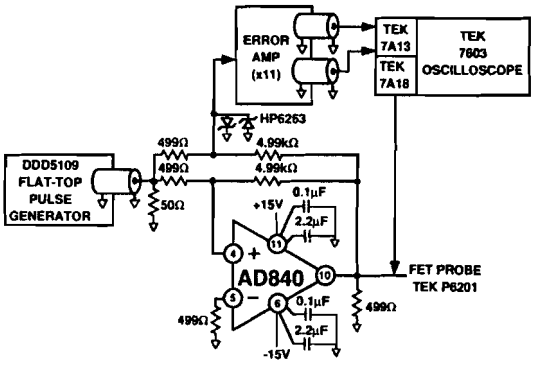


Figure 23. Settling Time Test Circuit

Figure 23 shows how measurement of the AD840's 0.01% settling in 100 ns was accomplished by amplifying the error signal from a false summing junction with a very high speed proprietary hybrid error amplifier specially designed to enable testing of small settling errors. The device under test was driving a 420 Ω load. The input to the error amp is clamped in order to avoid possible problems associated with the overdrive recovery of the oscilloscope input amplifier. The error amp amplifies the error from the false summing junction by 11, and it contains a gain vernier to fine trim the gain.

Figure 24 shows the "long-term" stability of the settling characteristics of the AD840 output after a 10 V step. There is no evidence of settling tails after the initial transient recovery time. The use of a junction isolated process, together with careful layout, avoids these problems by minimizing the effects of transistor isolation capacitance discharge and thermally induced shifts in circuit operating points. These problems do not occur even under high output current conditions.

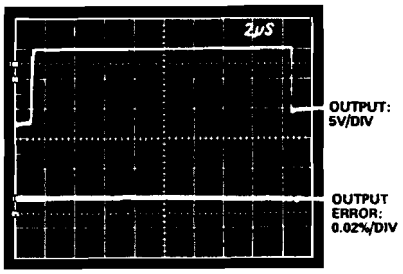


Figure 24. AD840 Settling Demonstrating No Settling Tails

GROUNDING AND BYPASSING

In designing practical circuits with the AD840, the user must remember that whenever high frequencies are involved, some special precautions are in order. Circuits must be built with short interconnect leads. Large ground planes should be used whenever possible to provide a low resistance, low inductance circuit path, as well as minimizing the effects of high frequency coupling. Sockets should be avoided, because the increased inter-lead capacitance can degrade bandwidth.

Feedback resistors should be of low enough value to assure that the time constant formed with the circuit capacitances will not limit the amplifier performance. Resistor values of less than 5 kΩ are recommended. If a larger resistor must be used, a small (±10 pF) feedback capacitor in connected parallel with the feedback resistor, R_F, may be used to compensate for these stray capacitances and optimize the dynamic performance of the amplifier in the particular application.

Power supply leads should be bypassed to ground as close as possible to the amplifier pins. A 2.2 μF capacitor in parallel with a 0.1 μF ceramic disk capacitor is recommended.

CAPACITIVE LOAD DRIVING ABILITY

Like all wideband amplifiers, the AD840 is sensitive to capacitive loading. The AD840 is designed to drive capacitive loads of up to 20 pF without degradation of its rated performance. Capacitive loads of greater than 20 pF will decrease the dynamic performance of the part although instability should not occur unless the load exceeds 100 pF. A resistor in series with the output can be used to decouple larger capacitive loads.

USING A HEAT SINK

The AD840 draws less quiescent power than most high speed amplifiers and is specified for operation without a heat sink. However, when driving low impedance loads the current to the load can be 4 to 5 times the quiescent current. This will create a noticeable temperature rise. Improved performance can be achieved by using a small heat sink such as the Aavid Engineering #602B.

HIGH SPEED DAC BUFFER CIRCUIT

The AD840's 100 ns settling time to 0.01% for a 10 V step makes it well suited as an output buffer for high speed D/A converters. Figure 25 shows the connections for producing a 0 to +10.24 V output swing from the AD568 35 ns DAC. With the AD568 in unbuffered voltage output mode, the AD840 is placed in noninverting configuration. As a result of the 1 kΩ span resistor provided internally in the AD568, the noise gain of this topology is 10. Only 5 pF is required across the feedback (span) resistor to optimize settling.

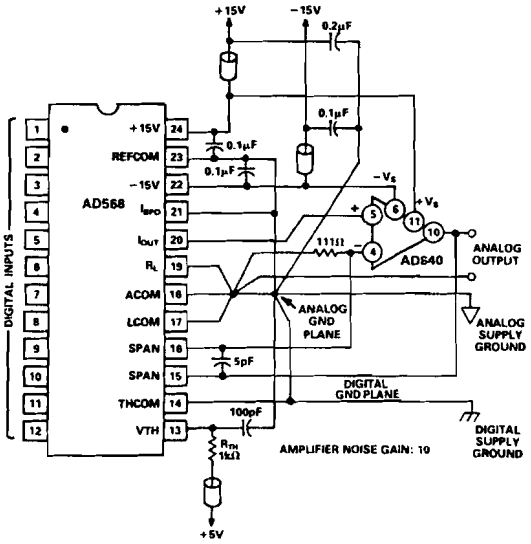


Figure 25. 0 to +10.24 V DAC Output Buffer

OVERDRIVE RECOVERY

Figure 26 shows the overdrive recovery capability of the AD840. Typical recovery time is 190 ns from negative overdrive and 350 ns from positive overdrive.

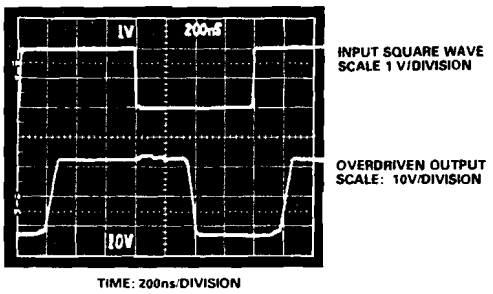


Figure 26. Overdrive Recovery

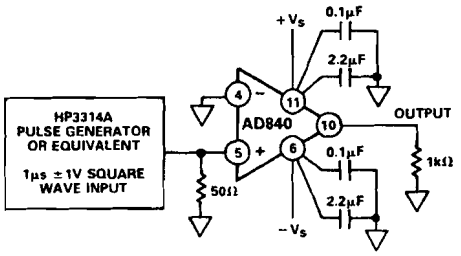


Figure 27. Overdrive Recovery Test Circuit

ANALOG DEVICES

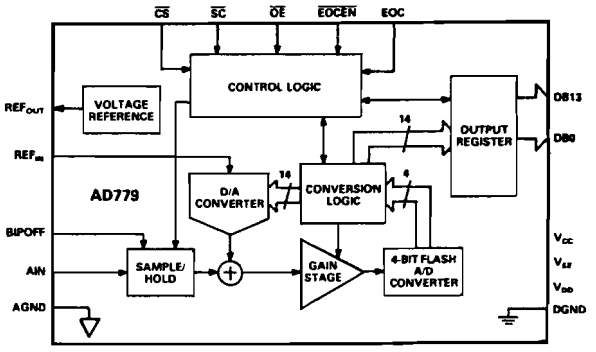
14-Bit 128 kSPS Complete Sampling ADC

AD779*

FEATURES

- **AC and DC Characterized and Specified (K, B, T Grades)**
- **128k Conversions per Second**
- **1 MHz Full Power Bandwidth**
- **500 kHz Full Linear Bandwidth**
- **80 dB S/N+D (K, B, T Grades)**
- **Twos Complement Data Format (Bipolar Mode)**
- **Straight Binary Data Format (Unipolar Mode)**
- **10 MΩ Input Impedance**
- **16-Bit Bus Interface (See AD679 for 8-Bit Interface)**
- **On-Board Reference and Clock**
- **10 V Unipolar or Bipolar Input Range**
- **MIL-STD-883 Compliant Versions Available**

FUNCTIONAL BLOCK DIAGRAM



PRODUCT DESCRIPTION

The AD779 is a complete, multipurpose 14-bit monolithic analog-to-digital converter, consisting of a sample-hold amplifier (SHA), a microprocessor compatible bus interface, a voltage reference and clock generation circuitry.

The AD779 is specified for ac (or "dynamic") parameters such as S/N+D ratio, THD and IMD which are important in signal processing applications. In addition, the AD779K, B and T grades are fully specified for dc parameters which are important in measurement applications.

The 14 data bits are accessed by a 16-bit bus in a single read operation. Data format is straight binary for unipolar mode and twos complement binary for bipolar mode. The input has a full-scale range of 10 V with a full power bandwidth of 1 MHz and a full linear bandwidth of 500 kHz. High input impedance (10 MΩ) allows direct connection to unbuffered sources without signal degradation.

This product is fabricated on Analog Devices' BiMOS process, combining low power CMOS logic with high precision, low noise bipolar circuits; laser-trimmed thin-film resistors provide high accuracy. The converter utilizes a recursive subranging algorithm which includes error correction and flash converter circuitry to achieve high speed and resolution.

The AD779 operates from +5 V and ±12 V supplies and dissipates 560 mW (typ). Twenty-eight-pin plastic DIP, ceramic DIP, and 44-J-leaded ceramic surface mount packages are available.

*Protected by U.S. Patent Numbers 4,804,960; 4,814,767; 4,833,345; 4,250,445; 4,808,908; RE30,586.

PRODUCT HIGHLIGHTS

1. COMPLETE INTEGRATION: The AD779 minimizes external component requirements by combining a high speed sample-hold amplifier (SHA), ADC, 5 V reference, clock and digital interface on a single chip. This provides a fully specified sampling A/D function unattainable with discrete designs.
2. SPECIFICATIONS: The AD779K, B and T grades provide fully specified and tested ac and dc parameters. The AD779J, A and S grades are specified and tested for ac parameters; dc accuracy specifications are shown as typicals. DC specifications (such as INL, gain and offset) are important in control and measurement applications. AC specifications (such as S/N+D ratio, THD and IMD) are of value in signal processing applications.
3. EASE OF USE: The pinout is designed for easy board layout, and the single cycle read output provides compatibility with 16-bit buses. Factory trimming eliminates the need for calibration modes or external trimming to achieve rated performance.
4. RELIABILITY: The AD779 utilizes Analog Devices' monolithic BiMOS technology. This ensures long term reliability compared to multichip and hybrid designs.
5. The AD779 is available in versions compliant with MIL-STD-883. Refer to the Analog Devices Military Products Databook or current AD779/883B data sheet for detailed specifications.

AD779—SPECIFICATIONS

AC SPECIFICATIONS (T_{min} to T_{max}, V_{CC} = +12 V ± 5%, V_{EE} = −12 V ± 5%, V_{DD} = +5 V ± 10%, f_{SAMPLE} = 128 kSPS, f_{IN} = 10.009 kHz unless otherwise noted)[1]

Parameter	AD779J/A/S Min	Typ	Max	AD779K/B/T Min	Typ	Max	Units
SIGNAL-TO-NOISE AND DISTORTION (S/N+D) RATIO[2]							
−0.5 dB Input (Referred to −0 dB Input)	78	79		80	81		dB
−20 dB Input (Referred to −20 dB Input)	58	59		60	61		dB
−60 dB Input (Referred to −60 dB Input)	18	19		20	21		dB
TOTAL HARMONIC DISTORTION (THD)[3]							
@ +25°C		−90	−84		−90	−84	dB
		0.003	0.006		0.003	0.006	%
T_{min} to T_{max}		−88	−82		−88	−82	dB
		0.004	0.008		0.004	0.008	%
PEAK SPURIOUS OR PEAK HARMONIC COMPONENT		−90	−84		−90	−84	dB
FULL POWER BANDWIDTH		1			1		MHz
FULL LINEAR BANDWIDTH	500			500			kHz
INTERMODULATION DISTORTION (IMD)[4]							
2nd Order Products		−90	−84		−90	−84	dB
3rd Order Products		−90	−84		−90	−84	dB

NOTES
[1]f_{IN} amplitude = −0.5 dB (9.44 V p-p) bipolar mode full scale unless otherwise indicated. All measurements referred to a −0 dB (9.997 V p-p) input signal unless otherwise noted.
[2]See Figure 15 for higher frequencies and other input amplitudes.
[3]See Figures 13 and 14 for higher frequencies and other input amplitudes.
[4]f_A = 9.08 kHz, f_B = 9.58 kHz, with f_{SAMPLE} = 128 kSPS. See Definition of Specifications section.
Specifications subject to change without notice.

DIGITAL SPECIFICATIONS (All device types T_{min} to T_{max}, V_{CC} = +12 V ± 5%, V_{EE} = −12 V ± 5%, V_{DD} = +5 V ± 10%)

Parameter		Test Conditions	Min	Max	Units
LOGIC INPUTS					
V_{IH}	High Level Input Voltage		2.0	V_{DD}	V
V_{IL}	Low Level Input Voltage		0	0.8	V
I_{IH}	High Level Input Current	V_{IN} = V_{DD}	−10	+10	μA
I_{IL}	Low Level Input Current	V_{IN} = 0 V	−10	+10	μA
C_{IN}	Input Capacitance			10	pF
LOGIC OUTPUTS					
V_{OH}	High Level Output Voltage	I_{OH} = 0.1 mA	4.0		V
		I_{OH} = 0.5 mA	2.4		V
V_{OL}	Low Level Output Voltage	I_{OL} = 1.6 mA		0.4	V
I_{OZ}	High Z Leakage Current	V_{IN} = V_{DD}	−10	+10	μA
C_{OZ}	High Z Output Capacitance			10	pF

NOTES
Specifications shown in **boldface** are tested on all devices at final electrical test with worst case supply voltages at T_{min}, +25°C and T_{max}. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in boldface are tested.
Specifications subject to change without notice.

DC SPECIFICATIONS (T_{min} to T_{max}, V_{CC} = +12 V ± 5%, V_{EE} = −12 V ± 5%, V_{DD} = +5 V ± 10% unless otherwise indicated)

Parameter	AD779J/A/S Min	AD779J/A/S Typ	AD779J/A/S Max	AD779K/B/T Min	AD779K/B/T Typ	AD779K/B/T Max	Units
TEMPERATURE RANGE							
J, K Grades	0		+70	0		+70	°C
A, B Grades	−40		+85	−40		+85	°C
S, T Grades	−55		+125	−55		+125	°C
ACCURACY							
Resolution	14			14			Bits
Integral Nonlinearity (INL)		±2			±1	±2	LSB
Differential Nonlinearity (DNL)	14			14			Bits
Unipolar Zero Error[1] (@ +25°C)		0.08			0.05	0.07	% FSR*
Bipolar Zero Error[1] (@ +25°C)		0.08			0.05	0.07	% FSR
Gain Error[1, 2] (@ +25°C)		0.12			0.09	0.11	% FSR
Temperature Drift							
Unipolar Zero[3]							
J, K Grades		0.04			0.04	0.05	% FSR
A, B Grades		0.05			0.05	0.07	% FSR
S, T Grades		0.09			0.09	0.10	% FSR
Bipolar Zero[3]							
J, K Grades		0.02			0.02	0.04	% FSR
A, B Grades		0.04			0.04	0.06	% FSR
S, T Grades		0.08			0.08	0.09	% FSR
Gain[3]							
J, K Grades		0.09			0.09	0.11	% FSR
A, B Grades		0.10			0.10	0.16	% FSR
S, T Grades		0.20			0.20	0.25	% FSR
Gain[4]							
J, K Grades		0.04			0.04	0.05	% FSR
A, B Grades		0.05			0.05	0.07	% FSR
S, T Grades		0.09			0.09	0.10	% FSR
ANALOG INPUT							
Input Ranges							
Unipolar Mode	0		+10	0		+10	V
Bipolar Mode	−5		+5	−5		+5	V
Input Resistance		10			10		MΩ
Input Capacitance		10			10		pF
Input Settling Time			1.5			1.5	μs
Aperture Delay		10			10		ns
Aperture Jitter		150			150		ps
INTERNAL VOLTAGE REFERENCE							
Output Voltage[5]	4.98		5.02	4.98		5.02	V
External Load							
Unipolar Mode			+1.5			+1.5	mA
Bipolar Mode			+0.5			+0.5	mA
POWER SUPPLIES							
Power Supply Rejection							
V_{CC} = +12 V ± 5%		±6				±6	LSB
V_{EE} = −12 V ± 5%		±6				±6	LSB
V_{DD} = +5 V ± 10%		±6				±6	LSB
Operating Current							
I_{CC}		18	20		18	20	mA
I_{EE}		25	34		25	34	mA
I_{DD}		8	12		8	12	mA
Power Consumption		560	745		560	745	mW

NOTES
[1]Adjustable to zero. See Figures 5 and 6.
[2]Includes internal voltage reference error.
[3]Includes internal voltage reference drift.
[4]Excludes internal voltage reference drift.
[5]With maximum external load applied.
*% FSR = percent of full-scale range.

Specifications shown in **boldface** are tested on all devices at final electrical test with worst case supply voltages at T_{min}, +25°C and T_{max}. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in boldface are tested.

Specifications subject to change without notice.

TIMING SPECIFICATIONS (All device types T_{min} to T_{max}, V_{CC} = +12 V ± 5%, V_{EE} = −12 V ± 5%, V_{DD} = +5 V ± 10%)

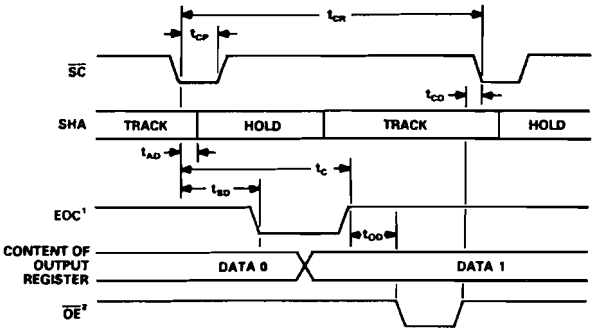
Parameter	Symbol	Min	Max	Units
Conversion Rate[1]	t_{CR}		7.8	µs
Convert Pulse Width	t_{CP}	97		ns
Aperture Delay	t_{AD}	5	20	ns
Conversion Time	t_C		6.3	µs
Status Delay	t_{SD}	0	400	ns
Access Time[2, 3]	t_{BA}	10	100	ns
		10	57[4]	ns
Float Delay[5]	t_{FD}	10	80	ns
Output Delay	t_{OD}		0	ns
$\overline{OE}$ Delay	t_{OE}	20		ns
Read Pulse Width	t_{RP}	100		ns
Conversion Delay	t_{CD}	400		ns

NOTES
[1]Includes Acquisition Time.
[2]Measured from the falling edge of $\overline{OE}/\overline{EOCEN}$ (0.8 V) to the time at which the data lines/EOC cross 2.0 V or 0.8 V. See Figure 4.
[3]C_{OUT} = 100 pF.
[4]C_{OUT} = 50 pF.
[5]Measured from the rising edge of $\overline{OE}/\overline{EOCEN}$ (2.0 V) to the time at which the output voltage changes by 0.5 V. See Figure 4; C_{OUT} = 10 pF.

Specifications shown in **boldface** are tested at final electrical test with worst case supply voltages at T_{min}, +25°C, and T_{max}. Results from those tests are used to calculate outgoing quality levels.

All min and max specifications are guaranteed although only those in boldface are tested.

Specifications subject to change without notice.



NOTES
[1]$\overline{EOCEN}$ = LOW.
[2]DATA SHOULD NOT BE ENABLED DURING A CONVERSION.

Figure 1. Conversion Timing

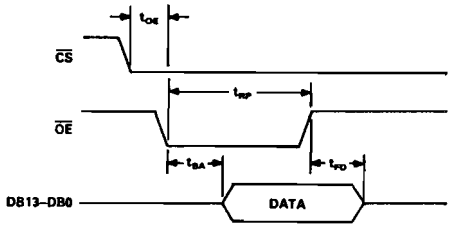


Figure 2. Output Timing

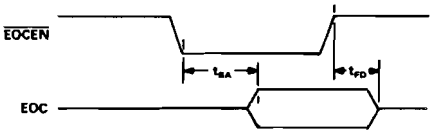


Figure 3. EOC Timing

TEST	V_{CP}	C_{OUT}
ACCESS TIME HIGH Z TO LOGIC LOW	5 V	100 pF
FLOAT TIME LOGIC HIGH TO HIGH Z	0 V	10 pF
ACCESS TIME HIGH Z TO LOGIC HIGH	0 V	100 pF
FLOAT TIME LOGIC LOW TO HIGH Z	5 V	10 pF

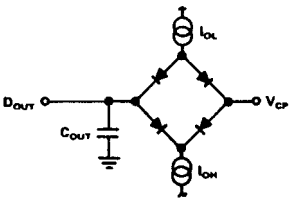


Figure 4. Load Circuit for Bus Timing Specifications

ABSOLUTE MAXIMUM RATINGS*

Specification	With Respect To	Min	Max	Units
V_{CC}	AGND	−0.3	+18	V
V_{EE}	AGND	−18	+0.3	V
V_{CC}	V_{EE}	−0.3	+26.4	V
V_{DD}	DGND	0	+7	V
AGND	DGND	−1	+1	V
AIN, REF_{IN}	AGND	V_{EE}	V_{CC}	V
Digital Inputs	DGND	−0.5	+7	V
Digital Outputs	DGND	−0.5	V_{DD} +0.3	V
Max Junction Temperature			175	°C
Operating Temperature				
J and K Grades		0	+70	°C
A and B Grades		−40	+85	°C
S and T Grades		−55	+125	°C
Storage Temperature		−65	+150	°C
Lead Temperature (10 sec max)			+300	°C

*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD SENSITIVITY

The AD779 features input protection circuitry consisting of large "distributed" diodes and polysilicon series resistors to dissipate both high energy discharges (Human Body Model) and fast, low energy pulses (Charged Device Model). Per Method 3015.2 of MIL-STD-883C, the AD779 has been classified as a Category 1 device.

Proper ESD precautions are strongly recommended to avoid functional damage or performance degradation. Charges as high as 4000 volts readily accumulate on the human body and test equipment and discharge without detection. Unused devices must be stored in conductive foam or shunts, and the foam should be discharged to the destination socket before devices are removed. For further information on ESD precautions, refer to Analog Devices' *ESD Prevention Manual*.

ORDERING GUIDE[1]

Model[2]	Package	Temperature Range	Tested and Specified	Package Option[3]
AD779JN	28-Pin Plastic DIP	0 to +70°C	AC	N-28A
AD779KN	28-Pin Plastic DIP	0 to +70°C	AC + DC	N-28A
AD779JD	28-Pin Ceramic DIP	0 to +70°C	AC	D-28A
AD779KD	28-Pin Ceramic DIP	0 to +70°C	AC + DC	D-28A
AD779AD	28-Pin Ceramic DIP	−40°C to +85°C	AC	D-28A
AD779BD	28-Pin Ceramic DIP	−40°C to +85°C	AC + DC	D-28A
AD779AJ	44-Lead Ceramic JLCC	−40°C to +85°C	AC	J-44
AD779BJ	44-Lead Ceramic JLCC	−40°C to +85°C	AC + DC	J-44
AD779SD	28-Pin Ceramic DIP	−55°C to +125°C	AC	D-28A
AD779TD	28-Pin Ceramic DIP	−55°C to +125°C	AC + DC	D-28A
AD779SJ	44-Lead Ceramic JLCC	−55°C to +125°C	AC	J-44
AD779TJ	44-Lead Ceramic JLCC	−55°C to +125°C	AC + DC	J-44

NOTES
[1]For two cycle read (8+16 bits) interface to 8-bit buses, see AD679.
[2]For details on grade and package offerings screened in accordance with MIL-STD-883, refer to the Analog Devices Military Products Databook or current AD779/883B data sheet.
[3]D = Ceramic DIP; J = J-Leaded Ceramic; N = Plastic DIP. For outline information see Package Information section.

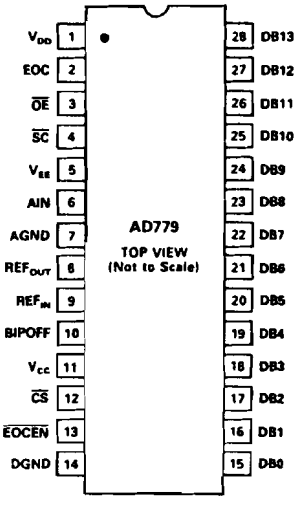
AD779 PIN DESCRIPTION

Symbol	28-Pin DIP Pin No.	44-Lead JLCC Pin No.	Type	Name and Function
AGND	7	11	P	Analog Ground. This is the ground return for AIN only.
AIN	6	10	AI	Analog Signal Input.
BIPOFF	10	15	AI	Bipolar Offset. Connect to AGND for +10 V input unipolar mode and straight binary output coding. Connect to REF_{OUT} for ±5 V input bipolar mode and twos-complement binary output coding.
$\overline{CS}$	12	19	DI	Chip Select. Active LOW.
DGND	14	23	P	Digital Ground
DB13–DB0	28–15	43, 42, 40, 39, 37, 36, 35, 34, 33, 31, 30, 27, 26, 25	DO	Data Bits. These pins provide all 14 bits in one 14 bit parallel output. Active HIGH
EOC	2	3	DO	End-of-Convert. EOC goes LOW when a conversion starts and goes HIGH when the conversion is finished. EOC is a three-state output. See $\overline{EOCEN}$ pin for information on EOC gating.
$\overline{EOCEN}$	13	21	DI	End-of-Convert Enable. Enables EOC pin. Active LOW.
$\overline{OE}$	3	5	DI	Output Enable. A down-going transition on $\overline{OE}$ enables data bits. Active LOW.
REF_{IN}	9	14	AI	Reference Input. +5 V input gives 10 V full scale range.
REF_{OUT}	8	12	AO	+5 V Reference Output. Tied to REF_{IN} for normal operation.
$\overline{SC}$	4	6	DI	Start Convert. Active LOW.
V_{CC}	11	17	P	+12 V Analog Power.
V_{EE}	5	8	P	−12 V Analog Power.
V_{DD}	1	1	P	+5 V Digital Power.

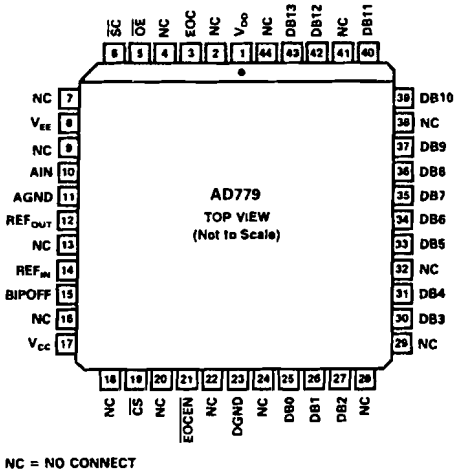
Type: AI = Analog Input.
AO = Analog Output.
DI = Digital Input.
DO = Digital Output. All DO pins are three-state drivers.
P = Power.

PIN CONFIGURATIONS

DIP Package



JLCC Package



NC = NO CONNECT

NYQUIST FREQUENCY
An implication of the Nyquist sampling theorem, the "Nyquist Frequency" of a converter is that input frequency which is one-half the sampling frequency of the converter.

SIGNAL-TO-NOISE AND DISTORTION (S/N+D) RATIO
S/N+D is the ratio of the rms value of the measured input signal to the rms sum of all other spectral components below the Nyquist frequency, including harmonics but excluding dc.

TOTAL HARMONIC DISTORTION (THD)
THD is the ratio of the rms sum of the first six harmonic components to the rms value of a full-scale input signal and is expressed as a percentage or in decibels. For input signals or harmonics that are above the Nyquist frequency, the aliased component is used.

PEAK SPURIOUS OR PEAK HARMONIC COMPONENT
The peak spurious or peak harmonic component is the largest spectral component excluding the input signal and dc. This value is expressed in decibels relative to the rms value of a full-scale input signal.

INTERMODULATION DISTORTION (IMD)
With inputs consisting of sine waves at two frequencies, fa and fb, any device with nonlinearities will create distortion products, of order (m + n), at sum and difference frequencies of mfa ± nfb, where m, n = 0, 1, 2, 3 . . . Intermodulation terms are those for which m or n is not equal to zero. For example, the second order terms are (fa + fb) and (fa − fb) and the third order terms are (2 fa + fb), (2 fa − fb), (fa + 2 fb) and (fa − 2 fb). The IMD products are expressed as the decibel ratio of the rms sum of the measured input signals to the rms sum of the distortion terms. The two signals applied to the converter are of equal amplitude and the peak value of their sum is −0.5 dB from full scale (9.44 V p–p). The IMD products are normalized to a 0-dB input signal.

BANDWIDTH
The full-power bandwidth is that input frequency at which the amplitude of the reconstructed fundamental is reduced by 3 dB for a full-scale input.

The full-linear bandwidth is the input frequency at which the slew rate limit of the sample-hold-amplifier (SHA) is reached. At this point, the amplitude of the reconstructed fundamental has degraded by less than −0.1 dB. Beyond this frequency, distortion of the sampled input signal increases significantly.

The AD779 has been designed to optimize input bandwidth, allowing it to undersample input signals with frequencies significantly above the converter's Nyquist frequency.

APERTURE DELAY
Aperture delay is a measure of the SHA's performance and is measured from the falling edge of Start Convert ($\overline{SC}$) to when the input signal is held for conversion.

APERTURE JITTER
Aperture jitter is the variation in aperture delay for successive samples and is manifested as noise on the input to the A/D.

INPUT SETTLING TIME
Settling time is a function of the SHA's ability to track fast slewing signals. This is specified as the maximum time required in track mode after a full-scale step input to guarantee rated conversion accuracy.

DIFFERENTIAL NONLINEARITY (DNL)
In an ideal ADC, code transitions are 1 LSB apart. Differential linearity is the deviation from this ideal value. It is often specified in terms of resolution for which no missing codes (NMC) are guaranteed.

INTEGRAL NONLINEARITY (INL)
The ideal transfer function for a linear ADC is a straight line drawn between "zero" and "full scale." The point used as "zero" occurs 1/2 LSB before the first code transition. "Full scale" is defined as a level 1 1/2 LSB beyond the last code transition. Integral nonlinearity error is the worst case deviation of a code from the straight line. The deviation of each code is measured from the middle of that code.

Note that the linearity error is not user adjustable.

POWER SUPPLY REJECTION
Variations in power supply will affect the full-scale transition, but not the converter's linearity. Power Supply Rejection is the maximum change in the full-scale transition point due to a change in power supply voltage from the nominal value.

TEMPERATURE DRIFT
This is the maximum change in the parameter from the initial value (@ +25°C) to the value at T_{min} or T_{max}.

UNIPOLAR ZERO ERROR
In unipolar mode, the first transition should occur at a level 1/2 LSB above analog ground. Unipolar zero error is the deviation of the actual transition from that point. This error can be adjusted as discussed in the Input Connections and Calibration section.

BIPOLAR ZERO ERROR
In the bipolar mode, the major carry transition (11 1111 1111 1111 to 00 0000 0000 0000) should occur at an analog value 1/2 LSB below analog ground. Bipolar zero error is the deviation of the actual transition from that point. This error can be adjusted as discussed in the Input Connections and Calibration section.

GAIN ERROR
The last transition should occur at an analog value 1 1/2 LSB below the nominal full scale (9.9991 volts for a 0–10 V range, 4.9991 volts for a ±5 V range). The gain error is the deviation of the actual level at the last transition from the ideal level with the zero error trimmed out. This error can be adjusted as shown in the Input Connections and Calibration section.

CONVERSION CONTROL

Before a conversion is started, End-of-Convert (EOC) is HIGH and the sample-hold is in track mode. A conversion is started by bringing $\overline{SC}$ LOW, regardless of the state of $\overline{CS}$.

After a conversion is started, the sample-hold goes into hold mode and EOC goes LOW, signifying that a conversion is in progress. During the conversion, the sample-hold will go back into track mode and start acquiring the next sample.

In track mode, the sample-hold will settle to ±0.003% (14 bits) in 1.5 μs maximum. The acquisition time does not affect the throughput rate as the AD779 goes back into track mode more than 2 μs before the next conversion. In multichannel systems, the input channel can be switched as soon as EOC goes LOW if the maximum throughput rate is needed.

When EOC goes HIGH, the conversion is completed and the output data may be read. Bringing $\overline{OE}$ LOW makes the output register contents available on the output data bits (DB13–DB0). A period of time t_{CD} is required after $\overline{OE}$ is brought HIGH before the next $\overline{SC}$ instruction is issued.

If $\overline{SC}$ is held LOW, conversion accuracy may deteriorate. For this reason, $\overline{SC}$ should not be held low in any attempt to operate in a continuously converting mode.

END-OF-CONVERT

End-of-Convert (EOC) is a three-state output which is enabled by End-of-Convert Enable $\overline{EOCEN}$.

OUTPUT ENABLE OPERATION

The data bits (DB13–DB0) are three-state outputs that are enabled by Chip Select ($\overline{CS}$) and Output Enable ($\overline{OE}$). $\overline{CS}$ should be LOW t_{OE} before $\overline{OE}$ is brought LOW. The output is read in a single cycle as a 14-bit word.

In unipolar mode (BIPOFF tied to AGND), the output coding is straight binary. In bipolar mode (BIPOFF tied to REF_{OUT}), output coding is twos complement binary.

POWER-UP

The AD779 typically requires 10 μs after power-up to reset internal logic.

14-BIT MODE CODING FORMAT (1 LSB = 0.61 mV)

Unipolar Coding (Straight Binary)		Bipolar Coding (Twos Complement)	
V_{IN}	Output Code	V_{IN}	Output Code
0.00000 V	000 . . . 0	−5.00000 V	100 . . . 0
5.00000 V	100 . . . 0	−0.00061 V	111 . . . 1
9.99939 V	111 . . . 1	0.00000 V	000 . . . 0
		+2.50000 V	010 . . . 0
		+4.99939 V	011 . . . 1

CONVERSION TRUTH TABLE

	INPUTS				OUTPUTS		
Mode	$\overline{SC}$	$\overline{EOCEN}$	$\overline{CS}$	$\overline{OE}$	EOC	DB13 . . . DB0	Status
Start Conversion	1	X	X	X			No Conversion
	↓	X	X	X			Start Conversion
	0	X	X	X			Continuous Conversion (Not Recommended)
Conversion Status	X	0	X	X	0		Converting
	X	0	X	X	1		Not Converting
	X	1	X	X	High Z		Either
Data Access	X	X	X	1		High Z	Three-State
	X	X	1	X		High Z	Three-State
	X	X	0	0		MSB . . . LSB	Data Out

NOTES
1 = HIGH voltage level.
0 = LOW voltage level.
X = Don't care.
↓ = HIGH to LOW transition. Must stay LOW for $t = t_{CP}$.

INPUT CONNECTIONS AND CALIBRATION

The high (10 MΩ) input impedance of the AD779 eases the task of interfacing to high source impedances or multiplexer channel-to-channel mismatches of up to 300 Ω. The 10 V p-p full scale input range accepts the majority of signal voltages without the need for voltage divider networks which could deteriorate the accuracy of the ADC.

The AD779 is factory trimmed to minimize offset, gain and linearity errors. In unipolar mode, the only external component that is required is a 50 Ω ±1% resistor. Two resistors are required in bipolar mode. If offset and gain are not critical, even these components can be eliminated.

In some applications, offset and gain errors need to be more precisely trimmed. The following sections describe the correct procedure for these various situations.

BIPOLAR RANGE INPUTS

The connections for the bipolar mode are shown in Figure 5. In this mode, data output coding will be twos complement binary. This circuit will allow approximately ±25 mV of offset trim range (±40 LSB) and ±0.5% of gain trim range (±80 LSB).

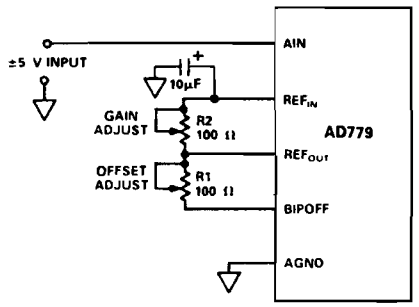


Figure 5. Bipolar Input Connections with Gain and Offset Trims

Either or both of the trim pots can be replaced with 50 Ω ±1% fixed resistors if the AD779 accuracy limits are sufficient for the application. If the pins are shorted together, the additional offset and gain errors will be approximately 80 LSB.

To trim bipolar zero to its nominal value, apply a signal 1/2 LSB below midrange (−0.305 mV for a ±5 V range) and adjust R1 until the major carry transition is located (11 1111 1111 1111 to 00 0000 0000 0000). To trim the gain, apply a signal 1 1/2 LSB below full scale (+4.9991 V for a ±5 V range) and adjust R2 to give the last positive transition (01 1111 1111 1110 to 01 1111 1111 1111). These trims are interactive so several iterations may be necessary for convergence.

A single pass calibration can be done by substituting a bipolar offset trim (error at minus full scale) for the bipolar zero trim (error at midscale), using the same circuit. First, apply a signal 1/2 LSB above minus full scale (−4.9997 V for a ±5 V range) and adjust R1 until the minus full scale transition is located (10 0000 0000 0000 to 10 000 000 0001). Then perform the gain error trim as outlined above.

UNIPOLAR RANGE INPUTS

Offset and gain errors can be trimmed out by using the configuration shown in Figure 6. This circuit allows approximately ±25 mV of offset trim range (±40 LSB) and ±0.5% of gain trim range (±80 LSB).

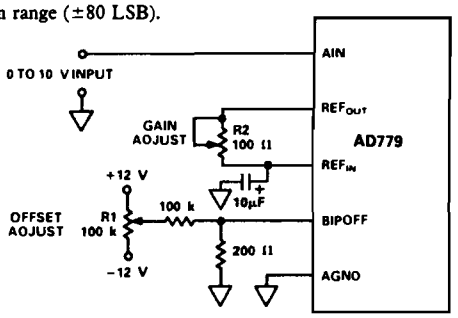


Figure 6. Unipolar Input Connections with Gain and Offset Trims

The first transition (from 00 0000 0000 0000 to 00 0000 0000 0001) should nominally occur for an input level of +1/2 LSB (0.305 mV above ground for a 10 V range). To trim unipolar zero to this nominal value, apply a 0.305 mV signal to AIN and adjust R1 until the first transition is located.

The gain trim is done by adjusting R2. If the nominal value is required, apply a signal 1 1/2 LSB below full scale (9.9997 V for a 10 V range) and adjust R2 until the last transition is located (11 1111 1111 1110 to 11 1111 1111 1111).

If offset adjustment is not required, BIPOFF should be connected directly to AGND. If gain adjustment is not required, R2 should be replaced with a fixed 50 Ω ±1% metal film resistor. If REF_{OUT} is connected directly to REF_{IN}, the additional gain error will be approximately 1%.

REFERENCE DECOUPLING

It is recommended that a 10 μF tantalum capacitor be connected between REF_{IN} (Pin 9) and ground. This has the effect of improving the S/N+D ratio through filtering possible broadband noise contributions from the voltage reference.

BOARD LAYOUT

Designing with high resolution data converters requires careful attention to board layout. Trace impedance is a significant issue. A 1.22 mA current through a 0.5 Ω trace will develop a voltage drop of 0.6 mV, which is 1 LSB at the 14-bit level for a 10 V full scale span. In addition to ground drops, inductive and capacitive coupling need to be considered, especially when high accuracy analog signals share the same board with digital signals. Finally, power supplies need to be decoupled in order to filter out ac noise.

Analog and digital signals should not share a common path. Each signal should have an appropriate analog or digital return routed close to it. Using this approach, signal loops enclose a small area, minimizing the inductive coupling of noise. Wide PC tracks, large gauge wire, and ground planes are highly recommended to provide low impedance signal paths. Separate analog and digital ground planes are also desirable, with a single interconnection point to minimize ground loops. Analog signals should be routed as far as possible from digital signals and should cross them at right angles.

The AD779 incorporates several features to help the user's layout. Analog pins (V_{EE}, AIN, AGND, REF_{OUT}, REF_{IN}, BIPOFF, V_{CC}) are adjacent to help isolate analog from digital signals. In addition, the 10 MΩ input impedance of AIN minimizes input trace impedance errors. Finally, ground currents have been minimized by careful circuit design. Current through AGND is 200 μA, with no code dependent variation. The current through DGND is dominated by the return current for DB13–DB0 and EOC.

SUPPLY DECOUPLING

The AD779 power supplies should be well filtered, well regulated, and free from high frequency noise. Switching power supplies are not recommended due to their tendency to generate spikes which can induce noise in the analog system.

Decoupling capacitors should be used as close as possible to all power supply pins. A 10 μF tantalum capacitor in parallel with a 0.1 μF ceramic capacitor provides adequate decoupling.

An effort should be made to minimize the trace length between the capacitor leads and the respective converter power supply and common pins. The circuit layout should attempt to locate the AD779, associated analog input circuitry and interconnections as far as possible from logic circuitry. A solid analog ground plane around the AD779 will isolate large switching ground currents. For these reasons, the use of wire wrap circuit construction is not recommended; careful printed circuit construction is preferred.

GROUNDING

If a single AD779 is used with separate analog and digital ground planes, connect the analog ground plane to AGND and the digital ground plane to DGND keeping lead lengths as short as possible. Then connect AGND and DGND together at the AD779. If multiple AD779s are used or the AD779 shares analog supplies with other components, connect the analog and digital returns together once at the power supplies rather than at each chip. This prevents large ground loops which inductively couple noise and allow digital currents to flow through the analog system.

USE OF EXTERNAL VOLTAGE REFERENCE

The AD779 features an on-chip voltage reference. For improved gain accuracy over temperature, a high performance external voltage reference may be used in place of the on-chip reference.

The AD586 and AD588 are popular references appropriate for use with high resolution converters. The AD586 is a low cost reference which utilizes a buried Zener architecture to provide low noise and drift. The AD588 is a higher performance reference which uses a proprietary ion-implanted buried Zener diode in conjunction with laser-trimmed thin-film resistors for low offset and low drift.

Figure 7 shows the use of the AD586 with the AD779 in a bipolar input mode. Over the 0 to +70°C range, the AD586 L-grade exhibits less than a 2.25 mV output change from its initial value at 25°C. REF_{IN} (Pin 9) scales its input by a factor of two; thus, this change becomes effectively 4.5 mV. When applied to the AD779, this results in a total gain drift of 0.09% FSR which is an improvement over the on-chip reference performance of 0.11% FSR. A noise-reduction capacitor, C_N, has been shown. This capacitor reduces the broadband noise of the AD586 output, thereby optimizing the overall ac and dc performance of the AD779.

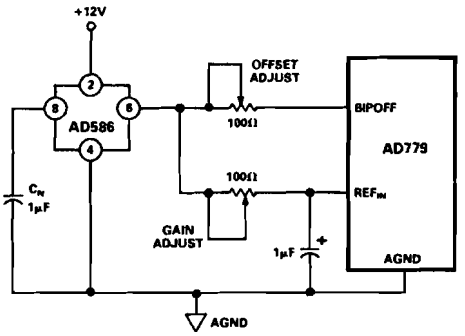


Figure 7. Bipolar Input with Gain and Offset Trims

Figure 8 shows the AD779 in unipolar input mode with the AD588 reference. The AD588 output is accurate to 0.65 mV from its value at 25°C over the 0 to 70°C range. This results in a 0.06% FSR total gain drift for the AD779, which is a substantial improvement over the on-chip reference performance of 0.11% FSR. A noise-reduction network on Pins 4, 6 and 7 has been shown. The 1 μF capacitors form low pass filters with the internal resistance of the AD588 Zener and amplifier cells and external resistance. This reduces the high frequency noise of the AD588, providing optimum ac and dc performance of the AD779.

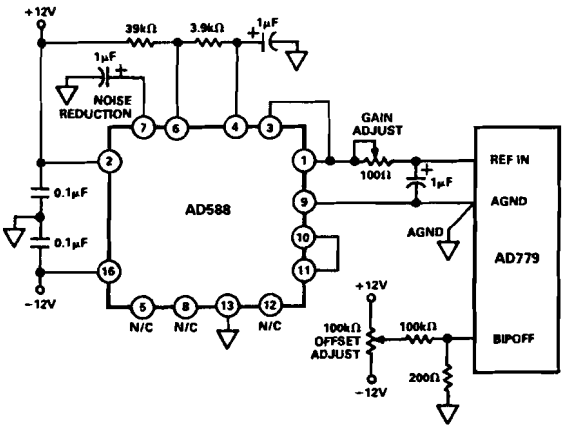


Figure 8. Unipolar Input with Gain and Offset Trims

INTERFACING THE AD779 TO MICROPROCESSORS

The I/O capabilities of the AD779 allow direct interfacing to general purpose and DSP microprocessor buses. The asynchronous conversion control feature allows complete flexibility and control with minimal external hardware.

The following examples illustrate typical AD779 interface configurations.

AD779 TO TMS320C25

In Figure 9 the AD779 is mapped into the TMS320C25 I/O space. AD779 conversions are initiated by issuing an OUT instruction to Port 1. EOC status and the conversion result are read in with an IN instruction to Port 1. A single wait state is inserted by generating the processor READY input from $\overline{IS}$, Port 1 and $\overline{MSC}$. This configuration supports processor clock speeds of 20 MHz and is capable of supporting processor clock speeds of 40 MHz if a NOP instruction follows each AD779 read instruction.

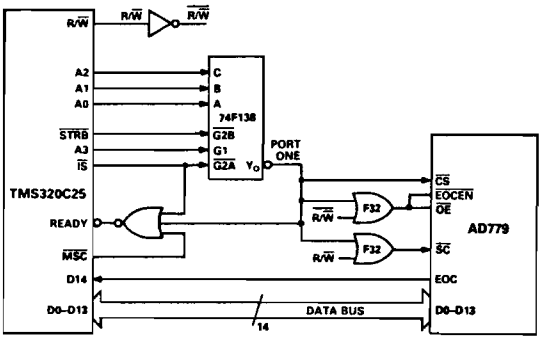


Figure 9. AD779 to TMS320C25 Interface

AD779 TO 80186

Figure 10 shows the AD779 interfaced to the 80186 microprocessor. This interface allows the 80186's built-in DMA controller to transfer the AD779 output into a RAM based FIFO buffer of any length, with no microprocessor intervention.

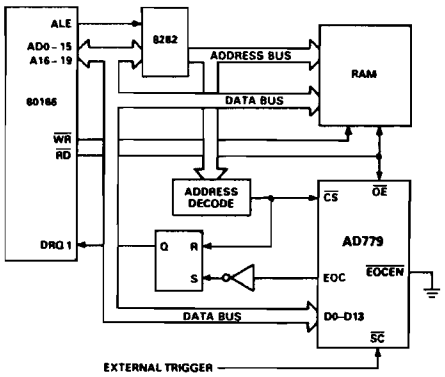


Figure 10. AD779 to 80186 DMA Interface

AD779 TO Z80

The AD779 can be interfaced to the Z80 processor in an I/O or memory mapped configuration. Figure 11 illustrates an I/O configuration, where the AD779 occupies several port addresses to allow separate polling of the EOC status and reading of the data. A useful feature of the Z80 is that a single wait state is automatically inserted during I/O operations, allowing the AD779 to be used with Z80 processors having clock speeds up to 8 MHz.

The AD779 is asynchronous which allows conversions to be initiated by an external trigger source independent of the microprocessor clock. After each conversion, the AD779 EOC signal generates a DMA request to Channel 1 (DRQ1). The subsequent DMA READ resets the interrupt latch. The system designer must assign a sufficient priority to the DMA channel to ensure that the DMA request will be serviced before the completion of the next conversion. This configuration can be used with 6 MHz and 8 MHz 80186 processors.

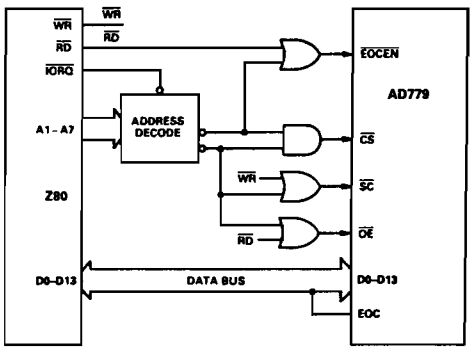


Figure 11. AD779 to Z80 Interface

AD779 TO ANALOG DEVICES ADSP-2100A

Figure 12 demonstrates the AD779 interfaced to an ADSP-2100A. With a clock frequency of 12.5 MHz, and instruction execution in one 80 ns cycle, the digital signal processor will support the AD779 data memory interface with two wait states.

The converter runs asychronously using a sampling clock. The EOC output to the AD779 gets asserted at the end of each conversion and causes an interrupt. Upon interrupt, the ADSP-2100A starts a data memory read by providing an address on the DMA bus. The decoded address generates $\overline{OE}$ for the converter. $\overline{OE}$, together with logic and latch, is used to force the ADSP-2100A into a one cycle wait state by generating DMACK. The read operation is thus started and completed within two processor cycles (160 ns).

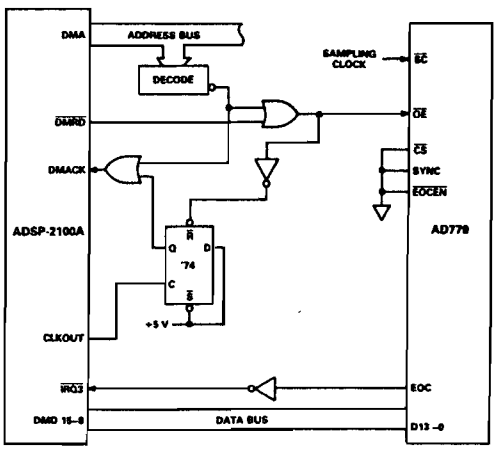


Figure 12. AD779 to ADSP-2100A Interface

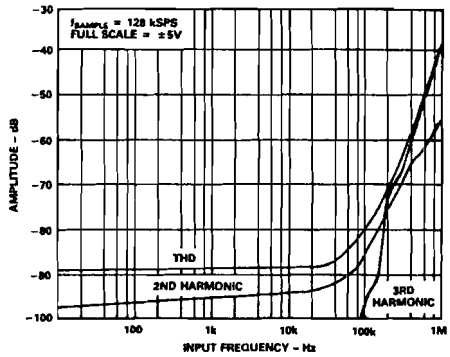


Figure 13. Harmonic Distortion vs. Input Frequency (−0.5 dB Input)

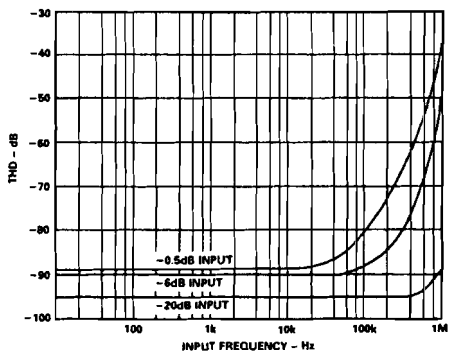


Figure 14. Total Harmonic Distortion vs. Input Frequency and Amplitude

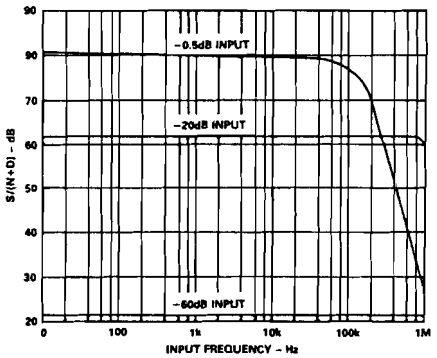


Figure 15. S/(N+D) vs. Input Frequency and Amplitude

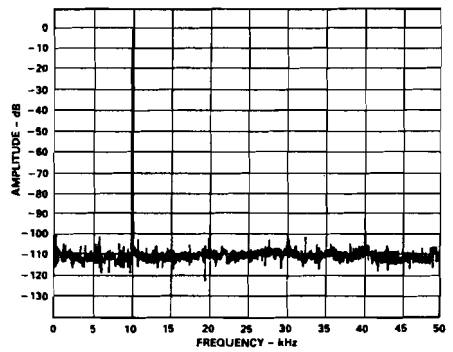


Figure 16. 5-Plot Averaged 2048-Point FFT at 128 kSPS, f_{IN} = 10.009 kHz

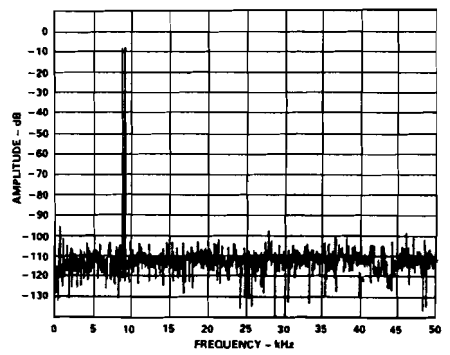


Figure 17. Nonaveraged IMD Plot for f_{IN} = 9.08 kHz (f_a), 9.58 kHz (f_b) at 128 kSPS

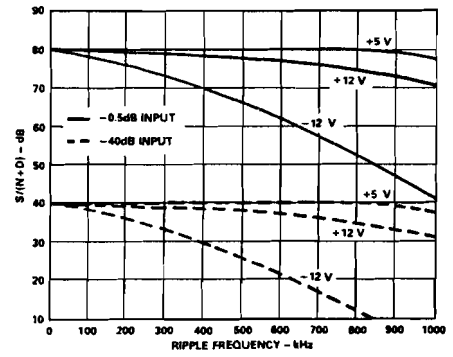


Figure 18. Power Supply Rejection (f_{IN} = 10 kHz, f_{SAMPLE} = 128 kSPS, V_{RIPPLE} = 0.1 V p–p)

Appendix 3: The Software Development System

The Software Development System (SWDS) is a PC-resident software development and debugging tool that provides real-time software simulation of the Texas Instruments TMS32020 and TMS320C25 digital signal processors. The SWDS packet consists of a hardware board and specific software.

The SWDS software consists of an assembler program, an archiver, a linker, and a debug-monitor program. The assembler translates assembly language source, which can be edited with a PC editor program, into machine language object files. The assembly language source files can contain instructions, assembler directives, and macro directives. The archiver allows the user to collect a group of files into a single archive file. One can for example use the archiver to collect a group of object files into an object library. The linker will include the members in the library that resolve external references during the link. The linker combines object files into a single executable object module. As it creates the executable module, it performs relocation and resolves external references. Linker directives allow the user to combine object file sections, bind sections or symbols to addresses or within memory ranges, and define or redefine global symbols.

The SWDS hardware consists of a single board designed around the PC I/O Extension Bus. The board can be plugged into a free expansion slot of a PC. It mainly contains a TMS320C25 (or TMS32020) device and 24k 16-bit words (zero wait-state) RAM memory.

The main purpose of the development process is to produce a program that can be executed in a TMS320C25 system. This is an electronic circuit based on a TMS320C25 device. The program can be executed from the debug-monitor program on the personal computer. The debug-monitor program provides full speed execution at 40 MHz, single-step execution, monitoring and editing registers and memory, and it provides the user to set up to 8 breakpoints. This environment provides the user to correct or refine the assembly language source, before executing it on the TMS320C25 system.

The programming procedure is as follows. After one or several assembly-language source files are translated into machine language object files, the linker program generates an executable file. This file is read into the RAM memory of the SWDS PC board. The PC board is connected to a 68-pins PGA-connector that is plugged into the IC socket of the

TMS320C25 on the TMS320C25 system. In this way, the operation of the TMS320C25 is emulated by the PC board.

The linker can generate output files in several standard formats. The SWDS software contains a converter program that can convert a Intel hexadecimal linker output file to binary format. This binary file can be loaded into EPROMs when the assembly language source has been corrected and refined.

Appendix 4: Assembly source of the transceiver program

```
;***************************************************************************
;
;  In this program, the single-dwell acquisition method is implemented.
;  The transceiver can be configured as a transmitter by setting DIP-switch
;  8 of S2 off. It can be configured as a receiver by setting DIP-switch 8
;  of S2 on. One transmitter and one receiver together with a
;  personal computer form a measurement system, that is controlled by the
;  personal computer. With this system, acquisition-time measurements can
;  be performed fully automatically for code lengths from 7 to 1023.
;  These measurements can be done for various spreading-code lengths
;  and various numbers of other users. The program can only be executed from
;  EPROM and not with the SWDS system, because the program allocates
;  more RAM than available on the SWDS system.
;
;***************************************************************************
;
;
;  Setting constants
;
;***************************************************************************

ROMB0       .set    0FF00h                  ; begin address of block B0 program memory
RAMB0       .set      200h                  ; begin address of block B0 data memory
RAMB1       .set      300h                  ; begin address of block B1
RAMB2       .set       60h                  ; begin address of block B2

TIM:        .set    2                       ; timer register
PRD:        .set    3                       ; timer period register
IMR:        .set    4                       ; interrupt mask register

AACK        .set    0Ch                     ; ATTENTION acknowledge  1100
DACK        .set    0Ah                     ; DATA acknowledge       1010
EOT         .set    06h                     ; End of transmition     0110
NACK        .set    03h                     ; Negative acknowledge   0011
SOH         .set    00h                     ; Start of header        0000

WAITLEN     .set     1875                   ; 1.5 e-4 sec at 50 MHz clock frequency
TMOUTL      .set    0FFFFh                  ; Timeout length = n*80 nsec
STACKPTR    .set       78h                  ; Stack pointer, end of RAMB2

;***************************************************************************
;
;  Setting the interrupt vectors
;
;***************************************************************************

        .sect "intvec"
        B       Begin                       ; Reset
        B       DAC_Int                     ; DAC interrupt
        B       ADC_Int                     ; ADC interrupt
        B       PAR_Int                     ; Parallel port interrupt
        .space  16*16
        B       Timer                       ; Timeout on communication
        B       X_Int                       ; Not used
        B       R_Int                       ; Not used
```

```
;***************************************************************************
;
; *** Configuring Block B1 on-chip RAM 300h-3FFh ***
;
;***************************************************************************

NIL             .usect  "RAMB1",1       ; NIL       = 0
ONE             .usect  "RAMB1",1       ; ONE       = 1
DIP             .usect  "RAMB1",1       ; memory location to store DIP-switch setting
OutData         .usect  "RAMB1",1       ; data to be sent to DAC
Amp             .usect  "RAMB1",1       ; help variable to scale DAC output
Amplitude       .usect  "RAMB1",1       ; help variable to scale DAC output
SignExt         .usect  "RAMB1",1       ; SignExt   = 8000h
Device          .usect  "RAMB1",1       ; transmitter = 0, receiver = 1
Count           .usect  "RAMB1",1       ; counter for various purposes
Help1           .usect  "RAMB1",1       ; help variable
Help2           .usect  "RAMB1",1       ; help variable

; *** Parallel port vars ***
AR0VAR          .usect  "RAMB1",1       ; memory location to save AR0
AR1VAR          .usect  "RAMB1",1       ; memory location to save AR1
AR3VAR          .usect  "RAMB1",1       ; memory location to save AR3
VAR1            .usect  "RAMB1",1       ; multi-purpose variable
VAR2            .usect  "RAMB1",1       ; multi-purpose variable
DATA            .usect  "RAMB1",1       ; data read from parallel port
TRDATA          .usect  "RAMB1",1       ; data to be sent to personal computer
TimeOut         .usect  "RAMB1",1       ; variable indicating number of time-outs
PAR_ZeroCnt     .usect  "RAMB1",1       ; variable used to check if first
                                        ; character sent is SOH
PAR_Status      .usect  "RAMB1",1       ; variable concerning the status of communications
                                        ; 0 = No communications, 1 = Receiving command
                                        ; 2 = Command avalable, 3 = Transmitting answer
PAR_RecLen      .usect  "RAMB1",1       ; length of message to receive
PAR_RecCnt      .usect  "RAMB1",1       ; number of characters received
PAR_TrmLen      .usect  "RAMB1",1       ; length of message to send
PAR_TrmCnt      .usect  "RAMB1",1       ; number of characters to send
PAR_Retry       .usect  "RAMB1",1       ; number of retries


; *** Generation of PN-sequences vars ***
AR4VARP         .usect  "RAMB1",1       ; memory location to save AR4
AR4VARM         .usect  "RAMB1",1       ; memory location to save AR4
AR3_1           .usect  "RAMB1",1       ; memory location to save AR3
AR3_2           .usect  "RAMB1",1       ; memory location to save AR3
AR3_3           .usect  "RAMB1",1       ; memory location to save AR3
AR3_4           .usect  "RAMB1",1       ; memory location to save AR3
AR3_5           .usect  "RAMB1",1       ; memory location to save AR3
AR3_6           .usect  "RAMB1",1       ; memory location to save AR3
AR3_7           .usect  "RAMB1",1       ; memory location to save AR3
AR3_8           .usect  "RAMB1",1       ; memory location to save AR3
AR3_9           .usect  "RAMB1",1       ; memory location to save AR3
AR3_10          .usect  "RAMB1",1       ; memory location to save AR3
AR3_11          .usect  "RAMB1",1       ; memory location to save AR3
AR3_12          .usect  "RAMB1",1       ; memory location to save AR3
AR3_13          .usect  "RAMB1",1       ; memory location to save AR3
AR3_14          .usect  "RAMB1",1       ; memory location to save AR3
AR3_15          .usect  "RAMB1",1       ; memory location to save AR3
AR3_16          .usect  "RAMB1",1       ; memory location to save AR3
SSCodeM         .usect  "RAMB1",1       ; starting address of inverted spreading code
SSCodeP         .usect  "RAMB1",1       ; starting address of spreading code
SSUser          .usect  "RAMB1",1       ; starting address of spreading codes
                                        ; of other users
SRLen           .usect  "RAMB1",1       ; shift register length
DataNumber      .usect  "RAMB1",1       ; number of bytes to receive
BitNumber       .usect  "RAMB1",1       ; number of bits to receive
SendLen         .usect  "RAMB1",1       ; length of message received in bytes
Bit             .usect  "RAMB1",1       ; last received bit
Phase           .usect  "RAMB1",1       ; starting phase of local spreading code
Seq             .usect  "RAMB1",1       ; contents of shift register
NewSeq          .usect  "RAMB1",1       ; new contents of shift register
```

```
RandomSeq       .usect  "RAMB1",1       ; contents of 16-bit shift register generating
                                        ; random signals to simulate other users
Mask            .usect  "RAMB1",1       ; mask for generation of random signals
Mask1           .usect  "RAMB1",1       ; mask for generation of random signals
Spare           .usect  "RAMB1",1       ; variable used for generation of random signals
Tap             .usect  "RAMB1",1       ; position of a feedback in the shift register
Tapx            .usect  "RAMB1",1       ; output of feedback shift register
Taps            .usect  "RAMB1",1       ; number of feedback taps in shift register
Len             .usect  "RAMB1",1       ; length of spreading code
Len3            .usect  "RAMB1",1       ; spreading-code length=7
Len4            .usect  "RAMB1",1       ; spreading-code length=15
Len5            .usect  "RAMB1",1       ; spreading-code length=31
Len6            .usect  "RAMB1",1       ; spreading-code length=63
Len7            .usect  "RAMB1",1       ; spreading-code length=127
Len8            .usect  "RAMB1",1       ; spreading-code length=255
Len9            .usect  "RAMB1",1       ; spreading-code length=511
Len10           .usect  "RAMB1",1       ; spreading-code length=1023
Users           .usect  "RAMB1",1       ; address pointing to spreading codes
                                        ; of other users
Hi_Corr         .usect  "RAMB1",1       ; high word of correlation value
Lo_Corr         .usect  "RAMB1",1       ; low word of correlation value
Hi_Acq          .usect  "RAMB1",1       ; high word of acquisition time
Lo_Acq          .usect  "RAMB1",1       ; low word of acquisition time
Hi_Th           .usect  "RAMB1",1       ; high word of threshold level
Lo_Th           .usect  "RAMB1",1       ; low word of threshold level
Hi_Time         .usect  "RAMB1",1       ; high word of maximum required time
Lo_Time         .usect  "RAMB1",1       ; low word of maximum required time
VerNr           .usect  "RAMB1",1       ; number of threshold exceedings occurred until now
VerifyNr        .usect  "RAMB1",1       ; number of threshold exceedings
Verification    .usect  "RAMB1",1       ; number of times the receiver entered
                                        ; a verification
OneCnt          .usect  "RAMB1",1       ; number of successive ones in message
LenCnt          .usect  "RAMB1",1       ; variable used to detect prefix of message
Data_Sync       .usect  "RAMB1",1       ; =1 if bit synchronization occurred
Frame_Sync      .usect  "RAMB1",1       ; =1 if frame synchronization occurred
Acq             .usect  "RAMB1",1       ; =1 if acquisition has been declared
Data_Bg         .usect  "RAMB1",1       ; pointer indication the beginning of the message
DataLen         .usect  "RAMB1",1       ; length of message to receive
Frame_Bg        .usect  "RAMB1",1       ; pointer indicating the beginning of a frame
ReqTime         .usect  "RAMB1",1       ; indicating maximum required acquisition time
Above           .usect  "RAMB1",1       ; =1 if maximum required time is exceeded
Temp1           .usect  "RAMB1",1       ; multi purpose variable
Temp2           .usect  "RAMB1",1       ; multi purpose variable

; *** ReadADC vars ***
InData          .usect  "RAMB1",1       ; temporary storage for IN instruction
EndOfADC_Int    .usect  "RAMB1",1       ; =1 if data is despread

; *** Parallel port ram ***
PAR_Answer      .usect  "DataRAM",512   ; memory section to store answer
PAR_TrmData     .usect  "DataRAM",512   ; memory section to store message to transmit
PAR_RecData     .usect  "DataRAM",256   ; memory section to store message received

;***************************************************************************
;
;  Parallel port possible commands and answers
;
;***************************************************************************

                .asect "Strings",0A000h
                .label  ParStrBeg

;***  Possible commands ***

CmdCnt          .word   4                       ; Number of commands to check
                .word   StopStr
                .word   DataStr
                .word   StartStr
                .word   ResetStr
```

```
StopStr         .word   Stop
                .string 0,  4, "Stop"           ; Check Stop
DataStr         .word   Data
                .string 0,  5, "Data "          ; Check Data
StartStr        .word   Start
                .string 0,  5, "Start"          ; Check Start
ResetStr        .word   Reset
                .string 0,  5, "Reset"          ; Check Reset


; *** Possible answers ***

DataRecStr      .string 0, 29, "Data received. Despreading..."
DataTrmStr      .string 0, 30, "Data received. Transmitting..."
UnknwCmdStr     .string 0, 15, "Unknown command"
OkStr           .string 0,  2, "Ok"
TooHighStr      .string 0, 25, "T_acq above required time"

                .label  ParStrEnd

;*****************************
;
; Main program start
;
;*****************************

                .text
Begin           DINT
                RXF
                CNFD                    ; Configure B0 as data RAM
                ROVM                    ; Reset overflow mode
                RSXM                    ; Reset set extension mode
                SPM     0               ; No shift on product register output
                LDPK    6
                LRLK    AR7,STACKPTR    ; Set software stack pointer

                ZAC
                SACL    OutData         ; DAC output voltage is +5.12 V
                OUT     OutData,2       ; to indicate program start

;***************************************************************************
;
;  Copy program from EPROM to RAM
;
;***************************************************************************

                LALK    copy_start      ; Help1 -> source pointer
                SACL    Help1
                LALK    8000h           ; Help2 -> destination pointer
                SACL    Help2
                LALK    abs_end-abs_code ; Temp1 -> program length
                SACL    Temp1

COPY            LAC     Help1
                TBLR    Temp2
                ADDK    1
                SACL    Help1

                LAC     Help2
                TBLW    Temp2
                ADDK    1
                SACL    Help2

                LAC     Temp1
                SUBK    1
                SACL    Temp1
                BGZ     COPY
```

```
;***********************************************************************
;
;  Main program initialization
;
;***********************************************************************

Beg_L1          LALK    0800h
                SACL    OutData             ; DAC output voltage is 0 V
                OUT     OutData,2           ; to indicate end of copy procedure

                LDPK    0
                LAC     IMR
                ANDK    0FFC0h
                ORK     4                   ; Enable parallel port interrupt
                SACL    IMR

                LALK    TMOUTL              ; Initialize TIM and PRD registers
                SACL    TIM
                SACL    PRD

                ZAC
                LARP    AR1
                LRLK    AR1,RAMB1           ; Clear RAMB1 and thus initialise
                RPTK    255                 ; all variables to zero
                SACL    *+
                LRLK    AR1,RAMB2           ; Clear RAMB2
                RPTK    31
                SACL    *+

                LRLK    AR1,RAMB0           ; RAMBLK0 data memory address
                RPTK    PrgLenBlk0
                BLKP    PrgBgBlk0,*+        ; Copy program into block B0
                CNFP                        ; Configure Block B0 as program memory

                LRLK    AR1,PAR_Answer      ; Copy PARALLEL port strings to RAM
                RPTK    ParStrEnd-ParStrBeg-1
                BLKP    ParStrBeg,*+

                LRLK    AR1,PNData          ; Copy pndata to RAM
                RPTK    SRDataLen
                BLKP    SRDataBg,*+

                LRLK    AR1,PNUserData      ; Copy pnuserdata to RAM
                RPTK    SRUserLen
                BLKP    SRUserDataBg,*+

                LDPK    6
                LACK    1
                SACL    ONE
                LACK    8
                SACL    BitNumber

                IN      VAR1,1              ; Clear parallel port D-FF and read dipswitches
                LAC     VAR1,5
                ANDK    3800h,5
                SACH    DIP

                LAC     VAR1
                ANDK    8000h
                BZ      TRINIT
RECINIT         LARP    AR1                 ; receiver initialization
                LACK    1
                SACL    Device

                B       CLRADCFF

TRINIT          ZAC                         ; transmitter initialization
                SACL    Device

CLRADCFF        CALL    ClrADC              ; Clear ADC Flipflops
```

```
PNSEQINIT       LDPK    6               ; initialization for the generation of
                LALK    8000h           ; PN-sequences
                SACL    Mask            ; set Mask
                LALK    8805h
                SACL    Mask1           ; set Mask1
                LALK    0FFFFh
                SACL    RandomSeq       ; set starting contents of shift register
                SACL    ReqTime         ; set maximum time to maximum

                LACK    07h             ; set length of pn sequences
                SACL    Len3
                LACK    0Fh
                SACL    Len4
                LACK    1Fh
                SACL    Len5
                LACK    3Fh
                SACL    Len6
                LACK    7Fh
                SACL    Len7
                LACK    0FFh
                SACL    Len8
                LALK    01FFh
                SACL    Len9
                LALK    03FFh
                SACL    Len10

                LRLK    AR3,SSCode10P   ; initialise all memory locations where
                LARP    AR3             ; PN sequences are stored to zero
                ZAC
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+

                LRLK    AR3,SSCode10M
                ZAC
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+

                LRLK    AR3,SSUser1
                ZAC
                RPT     Len8
                SACL    *+
                RPT     Len8
```

```
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser2
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser3
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser4
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser5
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser6
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser7
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
```

```
SACL    *+

LRLK    AR3,SSUser8
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser9
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser10
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser11
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser12
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser13
ZAC
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+
RPT     Len8
SACL    *+

LRLK    AR3,SSUser14
ZAC
```

```
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+

                LRLK    AR3,SSUser15
                ZAC
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+

                LRLK    AR3,SSUser16
                ZAC
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+
                RPT     Len8
                SACL    *+

                LRLK    AR3,SSUser1         ; save variables that point to
                SAR     AR3,AR3_1           ; start of memory sections
                LRLK    AR3,SSUser2
                SAR     AR3,AR3_2
                LRLK    AR3,SSUser3
                SAR     AR3,AR3_3
                LRLK    AR3,SSUser4
                SAR     AR3,AR3_4
                LRLK    AR3,SSUser5
                SAR     AR3,AR3_5
                LRLK    AR3,SSUser6
                SAR     AR3,AR3_6
                LRLK    AR3,SSUser7
                SAR     AR3,AR3_7
                LRLK    AR3,SSUser8
                SAR     AR3,AR3_8
                LRLK    AR3,SSUser9
                SAR     AR3,AR3_9
                LRLK    AR3,SSUser10
                SAR     AR3,AR3_10
                LRLK    AR3,SSUser11
                SAR     AR3,AR3_11
                LRLK    AR3,SSUser12
                SAR     AR3,AR3_12
                LRLK    AR3,SSUser13
                SAR     AR3,AR3_13
                LRLK    AR3,SSUser14
                SAR     AR3,AR3_14
                LRLK    AR3,SSUser15
                SAR     AR3,AR3_15
                LRLK    AR3,SSUser16
                SAR     AR3,AR3_16

COPYAMP         LARP    AR1                 ; Copy variables to scale DAC to RAM
                LRLK    AR1,Ampl0
                RPTK    1
                BLKP    Amp0,*+
                LRLK    AR1,Ampl1
                RPTK    2
```

```
                BLKP    Amp1,*+
                LRLK    AR1,Amp12
                RPTK    3
                BLKP    Amp2,*+
                LRLK    AR1,Amp13
                RPTK    4
                BLKP    Amp3,*+
                LRLK    AR1,Amp14
                RPTK    5
                BLKP    Amp4,*+
                LRLK    AR1,Amp15
                RPTK    6
                BLKP    Amp5,*+
                LRLK    AR1,Amp16
                RPTK    7
                BLKP    Amp6,*+
                LRLK    AR1,Amp17
                RPTK    8
                BLKP    Amp7,*+
                LRLK    AR1,Amp18
                RPTK    9
                BLKP    Amp8,*+
                LRLK    AR1,Amp19
                RPTK    10
                BLKP    Amp9,*+
                LRLK    AR1,Amp110
                RPTK    11
                BLKP    Amp10,*+
                LRLK    AR1,Amp111
                RPTK    12
                BLKP    Amp11,*+
                LRLK    AR1,Amp112
                RPTK    13
                BLKP    Amp12,*+
                LRLK    AR1,Amp113
                RPTK    14
                BLKP    Amp13,*+
                LRLK    AR1,Amp114
                RPTK    15
                BLKP    Amp14,*+
                LRLK    AR1,Amp115
                RPTK    16
                BLKP    Amp15,*+
                LRLK    AR1,Amp116
                RPTK    17
                BLKP    Amp16,*+

INITEND         LARP    AR5
                LALK    0FFFh
                SACL    OutData             ; DAC output voltage is -5.12 V
                OUT     OutData,2           ; to indicate end of initialization
                EINT

                B       MAINLOOP            ; end of initialization, branch to main program in
RAM

;***************************************************************************
;
;  Macros definitions
;
;***************************************************************************
;
; Save environment for interrupts (PAR_Int and Timer)
;
; Do not modify AR6-7 (Note: AR0 to AR5 are saved)
; Do not modify P and T register
;
;***************************************************************************
```

```
SaveEnv         $MACRO
                LARP    AR7
                MAR     *-
                SST1    *-              ; Save status registers
                SST     *-
                SACH    *-              ; Save accumulator
                SACL    *-
                SAR     AR0,*-          ; Save AR0 register
                SAR     AR1,*-          ; Save AR1 register
                SAR     AR2,*-          ; Save AR2 register
                SAR     AR3,*-          ; Save AR3 register
                SAR     AR4,*-          ; Save AR4 register
                SAR     AR5,*-          ; Save AR5 register
                $ENDM

;***************************************************************************
;
;  Restore environment for interrupts (PAR_Int and Timer)
;
;***************************************************************************

RestoreEnv      $MACRO
                LARP    AR7
                MAR     *+
                LAR     AR5,*+
                LAR     AR4,*+
                LAR     AR3,*+
                LAR     AR2,*+
                LAR     AR1,*+
                LAR     AR0,*+
                ZALS    *+              ; Restore accumulator
                ADDH    *+
                LST     *+              ; Save status registers (ook ARP)
                LST1    *+
                $ENDM

;*********************************************************
;
; Character to send to communications bus is constant
; Send 4 bit Character to parallel bus register
; If device is selected Character will be put on the bus
;
;*********************************************************

LptWrtC         $MACRO  Char
                LACK    :Char:
                SACL    TRDATA
                OUT     TRDATA,1        ; write data to communications bus
                ORK     10h
                SACL    TRDATA
                ZAC                     ; Set flag that timer interrupt is not timeout
                SACL    TimeOut
                LDPK    0               ; Switch to Memory based registers
                LALK    WAITLEN
                SACL    TIM
                LAC     IMR             ; Enable Timer interrupt
                ORK     8
                SACL    IMR
                LDPK    6
                $ENDM

;*********************************************************
;
; Send character in accumulator to communications bus
; Send 4 bit Character to parallel bus register
; If device is selected Character will be put on the bus
;
;*********************************************************
```

```
LptWrtA         $MACRO
                SACL    TRDATA
                OUT     TRDATA,1            ; write data to communications bus
                ORK     10h
                SACL    TRDATA
                ZAC                         ; Set flag that timer interrupt is not timeout
                SACL    TimeOut
                LDPK    0                   ; Switch to Memory based registers
                LALK    WAITLEN
                SACL    TIM
                LAC     IMR                 ; Enable Timer interrupt to generate interrupt
signal
                ORK     8
                SACL    IMR
                LDPK    6
                $ENDM


                .asect  "Prog",8000h        ; the program is executed from RAM
                .label  copy_start          ; the starting address is 8000h
abs_code

;***********************************************************************
;
;  Main program loop
;
;***********************************************************************

MAINLOOP        LAC     EndOfADC_Int        ; check if receiver has acquired correct
                BNZ     Despread_End        ; code phase
                LAC     Above               ; check if receiver has exceeded maximum
                BNZ     TooHigh             ; acquisition time
                LAC     PAR_Status          ; check if a command has been received
                SUBK    2
                BZ      PARCommand
                B       MAINLOOP

;*********************************************************************
;
; Respond to parallel port interrupt
; confirm reception or continue transmission depending on PAR_Status
;
;*********************************************************************

PAR_Int         DINT
                SaveEnv
                RSXM

                LARP    AR1
                LAR     AR1,AR1VAR          ; Restore AR1
                IN      DATA,1              ; Get Data from parallel interface
                LAC     DATA
                ANDK    00FFh               ; Remove dip-switch setting
                SACL    DATA

                BNZ     PAR_L1              ; Received character not SOH
                LAC     PAR_ZeroCnt
                ADDK    1                   ; Increment zero_count
                SACL    PAR_ZeroCnt
                SUBK    5                   ; Compare with 5
                BLZ     PAR_L2
                SACL    PAR_ZeroCnt         ; ACC = 0
                SACL    PAR_Status          ; PAR_Status = 0
                B       PAR_L2
PAR_L1          ZAC                         ; Reset zero count
                SACL    PAR_ZeroCnt

PAR_L2          LAC     PAR_Status          ; Test if PAR_Status=0
                BNZ     PAR_L3
                LAC     DATA                ; Get last received character
```

```
                BNZ     PAR_EndOfInt        ; Ignore everything except SOH
                LACK    1                   ; SOH received
                SACL    PAR_RecCnt          ; New status=1
                SACL    PAR_Status
                LRLK    AR1,PAR_RecData     ; Initialize receive buffer pointer
                LptWrtC AACK
                B       PAR_EndOfInt

PAR_L3          SUBK    1                   ; compare status = 1?
                BNZ     PAR_L10
                LAC     PAR_RecCnt
                SUBK    1                   ; ReceiveCount = 1?
                BNZ     PAR_L4

                LDPK    0                   ; Switch to register space
                LAC     IMR
                ANDK    0FFF0h              ; Disable parallel-port , Timer,
                                            ; transmit and receive interrupt
                SACL    IMR
                LDPK    6

                LAC     DATA
                SACL    PAR_RecLen
                LACK    2
                SACL    PAR_RecCnt
                LptWrtC DACK
                B       PAR_EndOfInt

PAR_L4          ADDK    2
                SUB     PAR_RecLen
                BGEZ    PAR_L7              ;PAR_RecCnt>=PAR_RecLen-1?
                LAC     DATA
                SACL    *+
                LAC     PAR_RecCnt
                ADDK    1
                SACL    PAR_RecCnt
                LptWrtC DACK
                B       PAR_EndOfInt

PAR_L7          BNZ     PAR_L8
                LAC     DATA
                SACL    *+
                LAC     PAR_RecCnt
                ADDK    1
                SACL    PAR_RecCnt
                LptWrtC EOT
                B       PAR_EndOfInt

PAR_L8          LAC     DATA
                SUBK    EOT
                BZ      PAR_L9
                ZAC
                SACL    PAR_Status
                SACL    PAR_RecCnt
                SACL    PAR_RecLen
                LptWrtC NACK                ; EOT expected
                B       PAR_EndOfInt

PAR_L9          LACK    2
                SACL    PAR_Status          ; Command received
                ZAC
                SACL    *                   ; Add end of string character
                B       PAR_EndOfInt

PAR_L10         SUBK    1                   ; Status = 2 ?
                BZ      PAR_EndOfInt        ; no character should have been received -> ignore

                LAC     PAR_TrmCnt          ; Status = 3
                SUBK    1
                BNZ     PAR_L12             ; TransmitCount = 1 ?
```

```
                LAC     DATA
                SUBK    AACK
                BNZ     PAR_L11             ; Is the received character AACK
                LAC     PAR_TrmCnt
                ADDK    1
                SACL    PAR_TrmCnt
                LARP    AR1
                LRLK    AR1,PAR_TrmData+1
                LAC     *+
                LptWrtA
                B       PAR_EndOfInt
PAR_L11         MAR     *-                  ; First character used as counter
                LAC     *
                ADDK    1
                SACL    *+
                SUBK    3                   ; Try three times without error
                BZ      PAR_RetryTrm
                LptWrtC SOH
                B       PAR_EndOfInt

PAR_L12         ADDK    1
                SUB     PAR_TrmLen
                BGEZ    PAR_L13             ; TransmCount < TransmLen ?
                LAC     DATA
                SUBK    DACK
                BNZ     PAR_RetryTrm        ; Transmitted DATA not acknowledged
                LAC     PAR_TrmCnt
                ADDK    1
                SACL    PAR_TrmCnt
                LAC     *+
                LptWrtA
                B       PAR_EndOfInt

PAR_L13         LAC     DATA
                SUBK    EOT
                BNZ     PAR_RetryTrm        ; No EOT received, any case retry!
                ZAC
                SACL    PAR_Status          ; Answer transmitted, everything OK!
                LptWrtC EOT
                B       PAR_EndOfInt

PAR_RetryTrm    LAC     PAR_Retry           ; Retry procedure
                ADD     ONE
                SACL    PAR_Retry
                SUBK    3                   ; Three retries
                BZ      PAR_L14             ; Give up ?
                ZAC
                LARP    AR1                 ; Needed if retry is invoked by Timeout
                LRLK    AR1,PAR_TrmData
                SACL    *+                  ; First character is set back to zero
                LACK    1
                SACL    PAR_TrmCnt
                LptWrtC SOH                 ; Try again
                B       PAR_EndOfInt
PAR_L14         SACL    PAR_Status          ; Three transmit retries failled, give up!

PAR_EndOfInt    SAR     AR1,AR1VAR          ; Save AR1
                LDPK    0                   ; Switch to register space
                LAC     IMR
                ORK     4                   ; Enable parallel interrupt, leave other
                                            ; interrupts as they are
                SACL    IMR
                LDPK    6
                RestoreEnv                  ; End Interrupt
                EINT
                RET
```

```
;*****************************************************************
;
;  A command has been received from the personal computer and an answer is found
;  This procedure sends the first character. The rest communication is interrupt driven.
;
;  On entry: AR0 points to answer STRING
;
;*****************************************************************

Transmit        LACK    3
                SACL    PAR_Status          ; make PAR_Status 3, indicating transmission
                ZAC                         ; of an answer
                SACL    PAR_Retry
                SACL    PAR_ZeroCnt

                LARP    AR1
                LRLK    AR1,PAR_TrmData
                SACL    *+,AR0              ; SOH

                LAR     AR2,*,AR2
                MAR     *-,AR0              ; Repeat count - 1
                LAC     *+,1,AR1
                ADDK    4
                SACL    PAR_TrmLen
                LAC     PAR_TrmLen,8
                SACH    *+
                ANDK    0F000h
                SACH    *+,4
                LAC     PAR_TrmLen
                ANDK    0Fh
                SACL    *+,AR0

TRM_L1          LAC     *,4,AR1
                SACH    *+                  ; Store High nibble
                ANDK    0F000h
                SACH    *+,4,AR2            ; Store Low nibble
                BANZ    TRM_L2,*-,AR0
                B       TRM_L3

TRM_L2          LAC     *+,12,AR1
                ANDK    0FF00h,4
                SACH    *+                  ; Store High nibble
                ANDK    0F000h
                SACH    *+,4,AR2            ; Store Low nibble
                BANZ    TRM_L1,*-,AR0

TRM_L3          LAC     ONE
                SACL    PAR_TrmCnt
                LptWrtC SOH                 ; Send SOH, first character of every transmit
                RET                         ; The rest is transmitted on interrupt basis

;*******************************************************
;
; Parallel PORT timer for interrupt signal generation and
; reception timeout detection
;
;*******************************************************

Timer           SaveEnv                     ; Interrupt : Timeout occured
                LDPK    6
                LAC     TimeOut
                BNZ     TI_L1
                OUT     TRDATA,1            ; Generate interrupt on acquisition computer
                LAC     ONE
                SACL    TimeOut
                LAC     PAR_Status
                LDPK    0
                BNZ     TI_L0
```

```
                LAC     IMR
                ANDK    0FFF7h              ; Disable Time-out interrupt
                SACL    IMR                 ; Last transmission completed, no answer expected
                LDPK    6
                RestoreEnv
                EINT
                RET

TI_L0           LALK    TMOUTL
                SACL    TIM
                LDPK    6
                RestoreEnv
                EINT
                RET

TI_L1           LDPK    0
                LAC     IMR
                ANDK    0FFF7h              ; Disable Time-out interrupt
                SACL    IMR
                LDPK    6
                LAC     PAR_Status
                SUBK    3
                BZ      PAR_RetryTrm        ; If transmitting, then retry
                ZAC
                SACL    PAR_Status
                RestoreEnv
                EINT
                RET


;*****************************************************************************************
;
; PARCommand routine
;
; Find subroutine for received command
; on entry PAR_RecData contains received command
;
; if no command is recognized "Unknown command" message is transmitted
;
; if command is recognized AR0 points to character after last checked character
; in receive buffer. (useful to retrieve data following command ID string)
;
;*****************************************************************************************

PARCommand      LARP    AR4
                LRLK    AR4,CmdCnt          ; Number of commands to check
                LAR     AR4,*
                MAR     *-,AR3
                LRLK    AR3,CmdCnt + 1      ; First command ID address

PC_L1           LAR     AR2,*+,AR2          ; AR2 points to command ID
                LAC     *+                  ; Subroutine address
                SACL    Temp2
                LAR     AR1,*+,AR1          ; Number of characters to check
                MAR     *-,AR2              ; AR1 = number of characters to check - 1
                LRLK    AR0,PAR_RecData

PC_L2           LAC     *,8,AR0             ; AR2 points to known command character
                SACH    Temp1               ; Save high byte
                LAC     *+,AR1              ; AR0 points to received character
                SUB     Temp1               ; compare with known character
                BNZ     PC_L4               ; equal ?
                BANZ    PC_L3,*-,AR2        ; equal yes, more characters to check?
                LARP    AR0
                LAC     Temp2               ; command found
                BACC                        ; goto subroutine

PC_L3           LAC     *+,AR0              ; AR2 points to known command character
                ANDK    0FFh                ; get low byte
                SUB     *+,AR1              ; compare with received character
```

```
                BNZ     PC_L4                   ; equal ?
                BANZ    PC_L2,*-,AR2            ; equal yes, more characters to check?
                LARP    AR0
                LAC     Temp2                   ; Command found
                BACC                            ; Goto subroutine

PC_L4           LARP    AR4
                BANZ    PC_L1,*-,AR3

                LRLK    AR0,UnknwCmdStr         ; Command not recognized!
                CALL    Transmit                ; Transmit "Unknown command"
                B       MAINLOOP


;***********************************************************************
;
;       Data is received and the data to send is stored in the sendbuffer
;
;       Interrupt 0 is enabled and transmission is started
;
;***********************************************************************

Data            SAR     AR0,AR0VAR

                LAC     Device
                BNZ     Data_L8

;******  transmitter has received data command ********

                LRLK    AR0,DataTrmStr          ; "Data received. Transmitting..."
                CALL    Transmit
Data_L1         LAC     PAR_Status              ; Do nothing, give timer chance to
                BNZ     Data_L1                 ; generate last interrupt of "Data
                RPTK    255                     ; received. Transmitting..."
                LAC     300h                    ; transmission
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h

                NOP

                DINT
                LAR     AR0,AR0VAR              ; AR0 points to data received from PC

                LAC     *+                      ; get data needed to generate the
                SACL    SRLen                   ; output signal

                ZAC
                LAC     *+
                ADD     *+,8
                SACL    Phase

                LAC     *+,AR1
                SACL    Users
```

```
                SAR     AR0,AR0VAR

                IN      InData,1                ; reset flip flop U20

                LRLK    AR1,Ampl0
                LAC     Users                   ; check how many other channel users
                BZ      Data_L3
                LRLK    AR1,Ampl1               ; set pointer to scaling data for the case
                SUBK    1                       ; of 1 other channel user
                BZ      Data_L3
                LRLK    AR1,Ampl2               ; .. for the case of 2 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl3               ; .. for the case of 3 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl4               ; .. for the case of 4 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl5               ; .. for the case of 5 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl6               ; .. for the case of 6 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl7               ; .. for the case of 7 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl8               ; .. for the case of 8 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl9               ; .. for the case of 9 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl10              ; .. for the case of 10 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl11              ; .. for the case of 11 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl12              ; .. for the case of 12 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl13              ; .. for the case of 13 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl14              ; .. for the case of 14 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl15              ; .. for the case of 15 other users
                SUBK    1
                BZ      Data_L3
                LRLK    AR1,Ampl16              ; .. for the case of 16 other users

Data_L3         SAR     AR1,Amplitude           ; Amplitude points to scaling data
                LAC     SRLen                   ; get shift-register length
                SUBK    3
                BZ      DIP0                    ; branch to procedures to generate spreading
                SUB     ONE                     ; code of the appropriate length
                BZ      DIP1                    ; DIP0 -> code length=7
                SUB     ONE                     ; DIP1 -> code length=15
                BZ      DIP2                    ; DIP2 -> code length=31
                SUB     ONE                     ; DIP3 -> code length=63
                BZ      DIP3                    ; DIP4 -> code length=127
                SUB     ONE                     ; DIP5 -> code length=255
                BZ      DIP4                    ; DIP6 -> code length=511
                SUB     ONE                     ; DIP7 -> code length=1023
                BZ      DIP5
                SUB     ONE
                BZ      DIP6
```

```
                SUB     ONE
                BZ      DIP7

;*** Generate PN-sequence of length 7 ***

DIP0            LRLK    AR3,PNData              ; AR3 -> place where info about
                LARP    3                       ;        SR and taps is stored
PN3             RTC
                LAR     AR0,Len3
                MAR     *+                      ; AR0 -> Len3
                LAR     AR2,*+                  ; AR2 -> # taps in shift reg.
                SAR     AR2,Taps
                LAR     AR5,NIL                 ; AR5 -> counter for chip nr.
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10M
                SAR     AR4,AR4VARM
                LRLK    AR4,SSCode10P           ; AR4 -> place where bits of
PN3LONG         CMPR    0                       ;        pn sequence are stored
                BBNZ    DIP0_1,*+,AR3
                LAR     AR2,Taps
PN3SHORT        LAR     AR1,*+,AR2              ; AR1 -> tap position of
                SAR     AR1,Tap                 ;        current tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1                  ; Tapx contains one of the feedback
                LAC     NewSeq                  ; taps. These are XORed with the other
                XOR     Tapx                    ; feedback taps, providing the new input
                SACL    NewSeq                  ; of the shift register which is stored
                MAR     *-                      ; into memory
                BANZ    PN3SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+,AR4
                LAC     *,15
                SFL
                ADD     NewSeq,13
                SACH    *+                      ; bits of sequence are stored
                SAR     AR4,AR4VARP
                LAR     AR4,AR4VARM
                XORK    8000h,1
                SACH    *+,AR5                  ; inverted bits of seq. are stored
                SAR     AR4,AR4VARM
                LAR     AR4,AR4VARP
                LAC     NewSeq
                ANDK    07h
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PN3LONG

DIP0_1          LAR     AR0,Len3                ; set pointers and variables of spreading
                SAR     AR0,Len                 ; code to be used
                LAR     AR1,ONE
                LRLK    AR2,SSCode10P
                SAR     AR2,SSCodeP
                LAC     SSCodeP
                ADD     Phase
                SACL    SSCodeP                 ; set phase of spreading code with regard to
                LRLK    AR2,SSCode10M           ; the codes of other users
                SAR     AR2,SSCodeM
                LAC     SSCodeM
                ADD     Phase
```

```
                SACL    SSCodeM
                LAR     AR2,SSCodeM
                LRLK    AR3,SSUser1
                SAR     AR3,SSUser
                RTC

                LRLK    AR2,7
                LRLK    AR4,SSCode10M
                LARP    AR5
                LRLK    AR5,SSCode10M
                ADRK    7
                LARP    AR4
COPY_PN3M       LAC     *+,AR5                  ; copy inverted spreading code again
                SACL    *+,AR2                  ; this provides the user to define an initial
                BANZ    COPY_PN3M,*-,AR4        ; phase without the need to reset pointers
                                                ; more than once within a whole sequence
                LRLK    AR2,7
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    7
                LARP    AR4
COPY_PN3P       LAC     *+,AR5                  ; copy spreading code again
                SACL    *+,AR2
                BANZ    COPY_PN3P,*-,AR4

                B       Data_L4

;*** Generate PN-sequence of length 15 ***      ; for comments see comments on
                                                ; generation of PN-sequence of length 7
DIP1            LRLK    AR3,PNData
                LARP    AR3
DIP1_0          LAC     *
                SUBK    4
                BZ      PN4
                CALL    SKIPCODE
                B       DIP1_0
PN4             RTC
                LAR     AR0,Len4
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10M
                SAR     AR4,AR4VARM
                LRLK    AR4,SSCode10P
PN4LONG         CMPR    0
                BBNZ    DIP1_1,*+,AR3
                LAR     AR2,Taps
PN4SHORT        LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PN4SHORT,*,AR3
                RPT     Taps
```

```
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,12
                SACH    *+
                SAR     AR4,AR4VARP
                LAR     AR4,AR4VARM
                XORK    8000h,1
                SACH    *+,AR5
                SAR     AR4,AR4VARM
                LAR     AR4,AR4VARP
                LAC     NewSeq
                ANDK    0Fh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PN4LONG

DIP1_1          LAR     AR0,Len4
                SAR     AR0,Len
                LAR     AR1,ONE
                LRLK    AR2,SSCode10P
                SAR     AR2,SSCodeP
                LAC     SSCodeP
                ADD     Phase
                SACL    SSCodeP
                LRLK    AR2,SSCode10M
                SAR     AR2,SSCodeM
                LAC     SSCodeM
                ADD     Phase
                SACL    SSCodeM
                LAR     AR2,SSCodeM
                LRLK    AR3,SSUser1
                SAR     AR3,SSUser

                LRLK    AR2,15
                LRLK    AR4,SSCode10M
                LARP    AR5
                LRLK    AR5,SSCode10M
                ADRK    15
                LARP    AR4
COPY_PN4M       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN4M,*-,AR4

                LRLK    AR2,15
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    15
                LARP    AR4
COPY_PN4P       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN4P,*-,AR4

                B       Data_L4

;*** Generate PN-sequence of length 31 ***      ; for comments see comments on
                                                ; generation of PN-sequence of length 7
DIP2            LRLK    AR3,PNData
                LARP    AR3
DIP2_0          LAC     *
                SUBK    5
                BZ      PN5
                CALL    SKIPCODE
                B       DIP2_0
PN5             RTC
                LAR     AR0,Len5
```

```
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10M
                SAR     AR4,AR4VARM
                LRLK    AR4,SSCode10P
PN5LONG         CMPR    0
                BBNZ    PNU5INIT,*+,AR3
                LAR     AR2,Taps
PN5SHORT        LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PN5SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,11
                SACH    *+
                SAR     AR4,AR4VARP
                LAR     AR4,AR4VARM
                XORK    8000h,1
                SACH    *+,AR5
                SAR     AR4,AR4VARM
                LAR     AR4,AR4VARP
                LAC     NewSeq
                ANDK    1Fh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PN5LONG

;*** Generation of the spreading code of length 31 of the other users ***

PNU5INIT        LAR     AR6,Users               ; AR6 -> number of users to go
                SAR     AR6,Count
                LRLK    AR3,PNUserData          ; AR3 -> place where info about
                LAR     AR0,Len5                ;        SR and feedback taps is stored
                B       PNU5                    ; AR0 -> Len5=31
PNU5_1          RPT     Taps
                MAR     *+
                MAR     *-
PNU5            RTC
                LARP    AR6
                BANZ    PNU5_2,*-,AR3
                B       DIP2_1
PNU5_2          MAR     *+
                LAR     AR2,*+                  ; AR2 -> number of taps in shift register
                SAR     AR2,Taps
                LAR     AR5,NIL                 ; AR5 -> counter indicating chip number
                LAC     *+,AR5
                SACL    Seq
```

```
                SFL
                SACL    NewSeq
                LAC     Count
                SUBK    1
                SACL    Count
                BZ      PNU51                   ; how many users to go?
                SUBK    1
                BZ      PNU52
                B       DIP2_1
PNU51           LRLK    AR4,SSUser1             ; spreading code of user 1 is stored
                B       PNU5LONG                ; in memory section starting
                                                ; at address SSUser1
PNU52           LRLK    AR4,SSUser2             ; spreading code of user 2 is stored
PNU5LONG        CMPR    0                       ; in memory section starting
                                                ; at address SSUser2
                BBNZ    PNU5_1,*+,AR3
                LAR     AR2,Taps
PNU5SHORT       LAR     AR1,*+,AR2              ; AR1 -> position of current tap
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1                  ; Tapx contains one of the feedback taps
                LAC     NewSeq                  ; these are XORed providing the new input
                XOR     Tapx                    ; of the shift register which is stored
                SACL    NewSeq                  ; in memory
                MAR     *-
                BANZ    PNU5SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,11
                SACH    *+,AR5
                LAC     NewSeq
                ANDK    1Fh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PNU5LONG

DIP2_1          LAR     AR0,Len5
                SAR     AR0,Len
                LAR     AR1,ONE
                LRLK    AR2,SSCode10P
                SAR     AR2,SSCodeP
                LAC     SSCodeP
                ADD     Phase
                SACL    SSCodeP
                LRLK    AR2,SSCode10M
                SAR     AR2,SSCodeM
                LAC     SSCodeM
                ADD     Phase
                SACL    SSCodeM
                LAR     AR2,SSCodeM
                LRLK    AR3,SSUser1
                SAR     AR3,SSUser
                RTC

                LRLK    AR2,31
                LRLK    AR4,SSCode10M
                LARP    AR5
                LRLK    AR5,SSCode10M
                ADRK    31
```

```
                LARP    AR4
COPY_PN5M       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN5M,*-,AR4

                LRLK    AR2,31
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    31
                LARP    AR4
COPY_PN5P       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN5P,*-,AR4

                B       Data_L4

;*** Generate PN-sequence of length 63 ***      ; for comments see comments on
                                                ; generation of PN-sequence of length 7
DIP3            LRLK    AR3,PNData
                LARP    AR3
DIP3_0          LAC     *
                SUBK    6
                BZ      PN6
                CALL    SKIPCODE
                B       DIP3_0
PN6             RTC
                LAR     AR0,Len6
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10M
                SAR     AR4,AR4VARM
                LRLK    AR4,SSCode10P
PN6LONG         CMPR    0
                BBNZ    PNU6INIT,*+,AR3
                LAR     AR2,Taps
PN6SHORT        LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PN6SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,10
                SACH    *+
                SAR     AR4,AR4VARP
                LAR     AR4,AR4VARM
                XORK    8000h,1
                SACH    *+,AR5
                SAR     AR4,AR4VARM
```

```
                LAR     AR4,AR4VARP
                LAC     NewSeq
                ANDK    3Fh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PN6LONG

;*** Generation of the spreading code of length 63 of the other users ***
;  for comments see the comments on the generation of spreading codes
;  of length 31 of the other users

PNU6INIT        LAR     AR6,Users
                SAR     AR6,Count
                LRLK    AR3,PNUserData
                LAR     AR0,Len6
                B       PNU6
PNU6_1          RPT     Taps
                MAR     *+
                MAR     *-
PNU6            RTC
                LARP    AR6
                BANZ    PNU6_2,*-,AR3
                B       DIP3_1
PNU6_2          MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LAC     Count
                SUBK    1
                SACL    Count
                BZ      PNU61
                SUBK    1
                BZ      PNU62
                B       DIP3_1
PNU61           LRLK    AR4,SSUser1
                B       PNU6LONG
PNU62           LRLK    AR4,SSUser2
PNU6LONG        CMPR    0
                BBNZ    PNU6_1,*+,AR3
                LAR     AR2,Taps
PNU6SHORT       LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PNU6SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,10
                SACH    *+,AR5
                LAC     NewSeq
                ANDK    3Fh
```

```
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PNU6LONG

DIP3_1          LAR     AR0,Len6
                SAR     AR0,Len
                LAR     AR1,ONE
                LRLK    AR2,SSCode10P
                SAR     AR2,SSCodeP
                LAC     SSCodeP
                ADD     Phase
                SACL    SSCodeP
                LRLK    AR2,SSCode10M
                SAR     AR2,SSCodeM
                LAC     SSCodeM
                ADD     Phase
                SACL    SSCodeM
                LAR     AR2,SSCodeM
                LRLK    AR3,SSUser1
                SAR     AR3,SSUser
                RTC

                LRLK    AR2,63
                LRLK    AR4,SSCode10M
                LARP    AR5
                LRLK    AR5,SSCode10M
                ADRK    63
                LARP    AR4
COPY_PN6M       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN6M,*-,AR4

                LRLK    AR2,63
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    63
                LARP    AR4
COPY_PN6P       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN6P,*-,AR4

                B       Data_L4

;*** Generate PN-sequence of length 127 ***     ; for comments see comments on
                                                ; generation of PN-sequence of length 7
DIP4            LRLK    AR3,PNData
                LARP    AR3
DIP4_0          LAC     *
                SUBK    7
                BZ      PN7
                CALL    SKIPCODE
                B       DIP4_0
PN7             RTC
                LAR     AR0,Len7
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10M
                SAR     AR4,AR4VARM
                LRLK    AR4,SSCode10P
PN7LONG         CMPR    0
                BBNZ    PNU7INIT,*+,AR3
                LAR     AR2,Taps
```

```
PN7SHORT        LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PN7SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,9
                SACH    *+
                SAR     AR4,AR4VARP
                LAR     AR4,AR4VARM
                XORK    8000h,1
                SACH    *+,AR5
                SAR     AR4,AR4VARM
                LAR     AR4,AR4VARP
                LAC     NewSeq
                ANDK    7Fh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PN7LONG

;*** Generation of the spreading code of length 127 of the other users ***
;  for comments see the comments on the generation of spreading codes
;  of length 31 of the other users

PNU7INIT        LAR     AR6,Users
                SAR     AR6,Count
                LRLK    AR3,PNUserData
                LAR     AR0,Len7
                B       PNU7
PNU7_1          RPT     Taps
                MAR     *+
                MAR     *-
PNU7            RTC
                LARP    AR6
                BANZ    PNU7_2,*-,AR3
                B       DIP4_1
PNU7_2          MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LAC     Count
                SUBK    1
                SACL    Count
                BZ      PNU71
                SUBK    1
                BZ      PNU72
                SUBK    1
                BZ      PNU73
                SUBK    1
                BZ      PNU74
```

```
                SUBK    1
                BZ      PNU75
                SUBK    1
                BZ      PNU76
                SUBK    1
                BZ      PNU77
                SUBK    1
                BZ      PNU78
                SUBK    1
                BZ      PNU79
                SUBK    1
                BZ      PNU710
                SUBK    1
                BZ      PNU711
                SUBK    1
                BZ      PNU712
                SUBK    1
                BZ      PNU713
                SUBK    1
                BZ      PNU714
                SUBK    1
                BZ      PNU715
                SUBK    1
                BZ      PNU716
                B       DIP4_1
PNU71           LRLK    AR4,SSUser1
                B       PNU7LONG
PNU72           LRLK    AR4,SSUser2
                B       PNU7LONG
PNU73           LRLK    AR4,SSUser3
                B       PNU7LONG
PNU74           LRLK    AR4,SSUser4
                B       PNU7LONG
PNU75           LRLK    AR4,SSUser5
                B       PNU7LONG
PNU76           LRLK    AR4,SSUser6
                B       PNU7LONG
PNU77           LRLK    AR4,SSUser7
                B       PNU7LONG
PNU78           LRLK    AR4,SSUser8
                B       PNU7LONG
PNU79           LRLK    AR4,SSUser9
                B       PNU7LONG
PNU710          LRLK    AR4,SSUser10
                B       PNU7LONG
PNU711          LRLK    AR4,SSUser11
                B       PNU7LONG
PNU712          LRLK    AR4,SSUser12
                B       PNU7LONG
PNU713          LRLK    AR4,SSUser13
                B       PNU7LONG
PNU714          LRLK    AR4,SSUser14
                B       PNU7LONG
PNU715          LRLK    AR4,SSUser15
                B       PNU7LONG
PNU716          LRLK    AR4,SSUser16
PNU7LONG        CMPR    0
                BBNZ    PNU7_1,*+,AR3
                LAR     AR2,Taps
PNU7SHORT       LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
```

```
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PNU7SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,9
                SACH    *+,AR5
                LAC     NewSeq
                ANDK    7Fh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PNU7LONG

DIP4_1          LAR     AR0,Len7
                SAR     AR0,Len
                LAR     AR1,ONE
                LRLK    AR2,SSCode10P
                SAR     AR2,SSCodeP
                LAC     SSCodeP
                ADD     Phase
                SACL    SSCodeP
                LRLK    AR2,SSCode10M
                SAR     AR2,SSCodeM
                LAC     SSCodeM
                ADD     Phase
                SACL    SSCodeM
                LAR     AR2,SSCodeM
                LRLK    AR3,SSUser1
                SAR     AR3,SSUser
                RTC

                LRLK    AR2,127
                LRLK    AR4,SSCode10M
                LARP    AR5
                LRLK    AR5,SSCode10M
                ADRK    127
                LARP    AR4
COPY_PN7M       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN7M,*-,AR4


                LRLK    AR2,127
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    127
                LARP    AR4
COPY_PN7P       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN7P,*-,AR4

                B       Data_L4

;*** Generate PN-sequence of length 255 ***     ; for comments see comments on
                                                ; generation of PN-sequence of length 7
DIP5            LRLK    AR3,PNData
                LARP    AR3
DIP5_0          LAC     *
                SUBK    8
                BZ      PN8
                CALL    SKIPCODE
                B       DIP5_0
PN8             RTC
```

```
                LAR     AR0,Len8
                LAC     *
                SUBK    8
                BNZ     PN9
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10M
                SAR     AR4,AR4VARM
                LRLK    AR4,SSCode10P
PN8LONG         CMPR    0
                BBNZ    PNU8INIT,*+,AR3
                LAR     AR2,Taps
PN8SHORT        LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PN8SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,8
                SACH    *+
                SAR     AR4,AR4VARP
                LAR     AR4,AR4VARM
                XORK    8000h,1
                SACH    *+,AR5
                SAR     AR4,AR4VARM
                LAR     AR4,AR4VARP
                LAC     NewSeq
                ANDK    0FFh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PN8LONG

;*** Generation of the spreading code of length 255 of the other users ***
;  for comments see the comments on the generation of spreading codes
;  of length 31 of the other users

PNU8INIT        LAR     AR6,Users
                SAR     AR6,Count
                LRLK    AR3,PNUserData
                LAR     AR0,Len8
                B       PNU8
PNU8_1          RPT     Taps
                MAR     *+
                MAR     *-
PNU8            RTC
                LARP    AR6
                BANZ    PNU8_2,*-,AR3
                B       DIP5_1
```

```
PNU8_2          MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LAC     Count
                SUBK    1
                SACL    Count
                BZ      PNU81
                SUBK    1
                BZ      PNU82
                SUBK    1
                BZ      PNU83
                SUBK    1
                BZ      PNU84
                SUBK    1
                BZ      PNU85
                SUBK    1
                BZ      PNU86
                SUBK    1
                BZ      PNU87
                B       DIP5_1
PNU81           LRLK    AR4,SSUser1
                B       PNU8LONG
PNU82           LRLK    AR4,SSUser2
                B       PNU8LONG
PNU83           LRLK    AR4,SSUser3
                B       PNU8LONG
PNU84           LRLK    AR4,SSUser4
                B       PNU8LONG
PNU85           LRLK    AR4,SSUser5
                B       PNU8LONG
PNU86           LRLK    AR4,SSUser6
                B       PNU8LONG
PNU87           LRLK    AR4,SSUser7
PNU8LONG        CMPR    0
                BBNZ    PNU8_1,*+,AR3
                LAR     AR2,Taps
PNU8SHORT       LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PNU8SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,8
                SACH    *+,AR5
                LAC     NewSeq
                ANDK    0FFh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PNU8LONG
```

```
DIP5_1          LAR     AR0,Len8
                SAR     AR0,Len
                LAR     AR1,ONE
                LRLK    AR2,SSCode10P
                SAR     AR2,SSCodeP
                LAC     SSCodeP
                ADD     Phase
                SACL    SSCodeP
                LRLK    AR2,SSCode10M
                SAR     AR2,SSCodeM
                LAC     SSCodeM
                ADD     Phase
                SACL    SSCodeM
                LAR     AR2,SSCodeM
                LRLK    AR3,SSUser1
                SAR     AR3,SSUser
                RTC

                LRLK    AR2,255
                LRLK    AR4,SSCode10M
                LARP    AR5
                LRLK    AR5,SSCode10M
                ADRK    255
                LARP    AR4
COPY_PN8M       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN8M,*-,AR4

                LRLK    AR2,255
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    255
                LARP    AR4
COPY_PN8P       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN8P,*-,AR4

                B       Data_L4

;*** Generate PN-sequence of length 511 ***     ; for comments see comments on
                                                ; generation of PN-sequence of length 7
DIP6            LRLK    AR3,PNData
                LARP    AR3
DIP6_0          LAC     *
                SUBK    9
                BZ      PN9
                CALL    SKIPCODE
                B       DIP6_0
PN9             RTC
                LAR     AR0,Len9
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10M
                SAR     AR4,AR4VARM
                LRLK    AR4,SSCode10P
PN9LONG         CMPR    0
                BBNZ    PNU9INIT,*+,AR3
                LAR     AR2,Taps
PN9SHORT        LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
```

```
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PN9SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,7
                SACH    *+
                SAR     AR4,AR4VARP
                LAR     AR4,AR4VARM
                XORK    8000h,1
                SACH    *+,AR5
                SAR     AR4,AR4VARM
                LAR     AR4,AR4VARP
                LAC     NewSeq
                ANDK    1FFh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PN9LONG

;*** Generation of the spreading code of length 511 of the other users ***
;  for comments see the comments on the generation of spreading codes
;  of length 31 of the other users

PNU9INIT        LAR     AR6,Users
                SAR     AR6,Count
                LRLK    AR3,PNUserData
                LAR     AR0,Len9
                B       PNU9
PNU9_1          RPT     Taps
                MAR     *+
                MAR     *-
PNU9            RTC
                LARP    AR6
                BANZ    PNU9_2,*-,AR3
                B       DIP6_1
PNU9_2          MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LAC     Count
                SUBK    1
                SACL    Count
                BZ      PNU91
                SUBK    1
                BZ      PNU92
                SUBK    1
                BZ      PNU93
                SUBK    1
                BZ      PNU94
                SUBK    1
                BZ      PNU95
                SUBK    1
                BZ      PNU96
                SUBK    1
```

```
                BZ      PNU97
                SUBK    1
                BZ      PNU98
                B       DIP6_1
PNU91           LRLK    AR4,SSUser1
                B       PNU9LONG
PNU92           LRLK    AR4,SSUser2
                B       PNU9LONG
PNU93           LRLK    AR4,SSUser3
                B       PNU9LONG
PNU94           LRLK    AR4,SSUser4
                B       PNU9LONG
PNU95           LRLK    AR4,SSUser5
                B       PNU9LONG
PNU96           LRLK    AR4,SSUser6
                B       PNU9LONG
PNU97           LRLK    AR4,SSUser7
                B       PNU9LONG
PNU98           LRLK    AR4,SSUser8
PNU9LONG        CMPR    0
                BBNZ    PNU9_1,*+,AR3
                LAR     AR2,Taps
PNU9SHORT       LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PNU9SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,7
                SACH    *+,AR5
                LAC     NewSeq
                ANDK    1FFh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PNU9LONG

DIP6_1          LAR     AR0,Len9
                SAR     AR0,Len
                LAR     AR1,ONE
                LRLK    AR2,SSCode10P
                SAR     AR2,SSCodeP
                LAC     SSCodeP
                ADD     Phase
                SACL    SSCodeP
                LRLK    AR2,SSCode10M
                SAR     AR2,SSCodeM
                LAC     SSCodeM
                ADD     Phase
                SACL    SSCodeM
                LAR     AR2,SSCodeM
                LRLK    AR3,SSUser1
                SAR     AR3,SSUser
                RTC
```

```
                LRLK    AR2,511
                LRLK    AR4,SSCode10M
                LARP    AR5
                LRLK    AR5,SSCode10M
                ADRK    255
                ADRK    255
                ADRK    1
                LARP    AR4
COPY_PN9M       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN9M,*-,AR4

                LRLK    AR2,511
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    255
                ADRK    255
                ADRK    1
                LARP    AR4
COPY_PN9P       LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN9P,*-,AR4

                B       Data_L4

;*** Generate PN-sequence of length 1023 ***     ; for comments see comments on
                                                 ; generation of PN-sequence of length 7
DIP7            LRLK    AR3,PNData
                LARP    AR3
DIP7_0          LAC     *
                SUBK    10
                BZ      PN10
                CALL    SKIPCODE
                B       DIP7_0
PN10            RTC
                LAR     AR0,Len10
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10M
                SAR     AR4,AR4VARM
                LRLK    AR4,SSCode10P
PN10LONG        CMPR    0
                BBNZ    PNU10INIT,*+,AR3
                LAR     AR2,Taps
PN10SHORT       LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PN10SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
```

```
                LAC     *,15
                SFL
                ADD     NewSeq,6
                SACH    *+
                SAR     AR4,AR4VARP
                LAR     AR4,AR4VARM
                XORK    8000h,1
                SACH    *+,AR5
                SAR     AR4,AR4VARM
                LAR     AR4,AR4VARP
                LAC     NewSeq
                ANDK    3FFh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PN10LONG

;*** Generation of the spreading code of length 1023 of the other users ***
;  for comments see the comments on the generation of spreading codes
;  of length 31 of the other users

PNU10INIT       LAR     AR6,Users
                SAR     AR6,Count
                LRLK    AR3,PNUserData
                LAR     AR0,Len10
                B       PNU10
PNU10_1         RPT     Taps
                MAR     *+
                MAR     *-
PNU10           RTC
                LARP    AR6
                BANZ    PNU10_2,*-,AR3
                B       DIP7_1
PNU10_2         MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LAC     Count
                SUBK    1
                SACL    Count
                BZ      PNU101
                SUBK    1
                BZ      PNU102
                SUBK    1
                BZ      PNU103
                SUBK    1
                BZ      PNU104
                SUBK    1
                BZ      PNU105
                B       DIP7_1
PNU101          LRLK    AR4,SSUser1
                B       PNU10LONG
PNU102          LRLK    AR4,SSUser2
                B       PNU10LONG
PNU103          LRLK    AR4,SSUser3
                B       PNU10LONG
PNU104          LRLK    AR4,SSUser4
                B       PNU10LONG
PNU105          LRLK    AR4,SSUser5
PNU10LONG       CMPR    0
                BBNZ    PNU10_1,*+,AR3
                LAR     AR2,Taps
PNU10SHORT      LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
```

```
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    PNU10SHORT,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+
                LARP    AR4
                LAC     *,15
                SFL
                ADD     NewSeq,6
                SACH    *+,AR5
                LAC     NewSeq
                ANDK    3FFh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       PNU10LONG

DIP7_1          LAR     AR0,Len10
                SAR     AR0,Len
                LAR     AR1,ONE
                LRLK    AR2,SSCode10P
                SAR     AR2,SSCodeP
                LAC     SSCodeP
                ADD     Phase
                SACL    SSCodeP
                LRLK    AR2,SSCode10M
                SAR     AR2,SSCodeM
                LAC     SSCodeM
                ADD     Phase
                SACL    SSCodeM
                LAR     AR2,SSCodeM
                LRLK    AR3,SSUser1
                SAR     AR3,SSUser
                RTC

                LRLK    AR2,1023
                LRLK    AR4,SSCode10M
                LARP    AR5
                LRLK    AR5,SSCode10M
                ADRK    255
                ADRK    255
                ADRK    255
                ADRK    255
                ADRK    3
                LARP    AR4
COPY_PN10M      LAC     *+,AR5
                SACL    *+,AR2
                BANZ    COPY_PN10M,*-,AR4

                LRLK    AR2,1023
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    255
                ADRK    255
                ADRK    255
                ADRK    255
                ADRK    3
                LARP    AR4
COPY_PN10P      LAC     *+,AR5
                SACL    *+,AR2
```

```
                BANZ    COPY_PN10P,*-,AR4

Data_L4         LRLK    AR1,SendBufData         ; AR1 -> memory section where
                LAR     AR0,AR0VAR              ;        message is stored
                LARP    AR0                     ; AR0 -> message received from PC
                LDPK    6
                ZAC
                SACL    Count
                SACL    SendLen
Data_L5         LAC     *                       ; determine the length of the message
                SUBK    46                      ; to be transmitted; last character of message
                BZ      Data_L6                 ; is char(46), which is not transmitted
                LAC     Count
                ADDK    1
                SACL    Count
                LAC     SendLen
                ADDK    8
                SACL    SendLen
                LAC     *+,AR1
                SACL    *+,AR0
                B       Data_L5

Data_L6         LAC     Count
                SACL    DataNumber
                LRLK    AR1,SendBufData
                LAR     AR2,DataNumber
                LAR     AR3,BitNumber
                LAC     BitNumber
                SACL    Bit                     ; AR1 -> number of bytes to transmit
                LRLK    AR4,SendData            ; AR2 ->
                LARP    AR4                     ; AR3 and Bit points to bit in byte
                ADRK    7                       ; AR4 -> points to memory place
                LARP    AR3                     ;        where bits are stored

Data_L7         LT      Bit                     ; store message to transmit
                MAR     *-,AR1                  ; bit-wise in memory; bit determines
                LAC     Bit                     ; the bit which is stored into memory
                SUBK    1
                SACL    Bit
                LACT    *,AR4
                ANDK    0100h
                SFL
                SACH    *-,7,AR3
                BANZ    Data_L7,*
                LARP    AR1
                MAR     *+,AR2
                MAR     *-,AR4
                LAR     AR3,BitNumber
                LAC     BitNumber
                SACL    Bit
                ADRK    10h
                LARP    AR2
                BANZ    Data_L7,*,AR3

                LAC     Users
                SACL    Count
                LAR     AR0,Len                 ; AR0 -> length of spreading code
                LAR     AR1,ONE                 ; AR1 -> chip counter
                LAR     AR2,SSCodeM             ; AR2 -> place where code is stored
                LAR     AR3,SSUser              ; AR3 -> place where other codes are stored
                LRLK    AR4,SendData            ; AR4 -> place where message to send is stored
                LAR     AR5,SendLen             ; AR5 -> length of message
                LARP    AR5

                LDPK    0
                LAC     IMR
                ORK     01h
                SACL    IMR                     ; enable DAC_Interrupt
                LDPK    6
                EINT
```

```
                B       MAINLOOP

SKIPCODE        MAR     *+                      ; look for other data in memory
                LAC     *+                      ; used for the generation of a
                SACL    Taps                    ; spreading code
                RPT     Taps
                MAR     *+
                RET

;********* receiver has received a data command *****

Data_L8         LRLK    AR0,DataRecStr          ; "Data received. Transmitting..."
                CALL    Transmit
Data_L9         LAC     PAR_Status              ; Do nothing, give timer chance to
                BNZ     Data_L9                 ; generate last interrupt of "Data
                RPTK    255                     ; received. Transmitting..."
                LAC     300h                    ; transmission
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h

                NOP

                DINT

                LAR     AR0,AR0VAR              ; get data needed to generate
                LAC     *+                      ; local spreading code and
                SACL    SRLen                   ; to acquire the correct
                LAC     *+,8                    ; code phase
                ADD     *+
                SACL    Hi_Th
                LAC     *+,8
                ADD     *+
                SACL    Lo_Th
                LAC     *+
                SACL    VerifyNr
                LAC     *+,8
                ADD     *+
                SACL    ReqTime
                LAC     *+
                ADD     *+,8
                SACL    Phase

                IN      InData,1                ; reset flip flop U20

                LAC     SRLen                   ; check register length
                SUBK    3
                BZ      Rec3
                SUB     ONE
                BZ      Rec4
                SUB     ONE
                BZ      Rec5
                SUB     ONE
                BZ      Rec6
                SUB     ONE
```

```
                BZ      Rec7
                SUB     ONE
                BZ      Rec8
                SUB     ONE
                BZ      Rec9
                SUB     ONE
                BZ      Rec10

;*** Generation of the local spreading code of length 7 and ***
;***   initialization for the acquisition-time measurement  ***

Rec3            LRLK    AR3,PNData              ; AR3 -> place where info about
                LARP    AR3                     ;        SR and taps is stored
Rec3_1          RTC
                LAR     AR0,Len3                ; AR0 -> Len3
                MAR     *+
                LAR     AR2,*+                  ; AR2 -> # taps in shift register
                SAR     AR2,Taps
                LAR     AR5,NIL                 ; AR5 -> counter for chip number
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10P           ; AR4 -> place where pnsequence
Rec3_2          CMPR    0                       ;        is stored
                BBNZ    Rec3_5,*+,AR3
                LAR     AR2,Taps
Rec3_3          LAR     AR1,*+,AR2              ; AR1 -> tap position of
                SAR     AR1,Tap                 ;        current tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1                  ; Tapx contains one of the feedback
                LAC     NewSeq                  ; taps. These are XORed with the other
                XOR     Tapx                    ; feedback taps, providing the new input
                SACL    NewSeq                  ; of the shift register which is stored
                MAR     *-                      ; into memory
                BANZ    Rec3_3,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+,AR4
                LAC     *,15
                SFL
                ADD     NewSeq,13
                SACH    *                       ; new chip in stored in memory
                LAC     *
                BNZ     Rec3_4
                LACK    -1
                SACL    *
Rec3_4          MAR     *+                      ; determine new contents of shift register
                LARP    AR5
                LAC     NewSeq
                ANDK    07h
                SACL    Seq
                SFL
                SACL    NewSeq
                B       Rec3_2

Rec3_5          LRLK    AR2,7
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    7
                LARP    AR4
Rec3_6          LAC     *+,AR5                  ; copy spreading code again
                SACL    *+,AR2                  ; this provides the user to define an initial
```

```
                BANZ    Rec3_6,*-,AR4           ; phase without the need to reset pointers
                                                ; more than once within a whole sequence
Rec3_7          LAR     AR0,Len3
                SAR     AR0,Len
                LALK    SSCode10P
                ADD     Phase                   ; determine initial phase of
                SACL    SSCodeP                 ; the local spreading code
                LAR     AR1,SSCodeP
                LAR     AR2,ONE

                B       Corr_Init

;*** Generation of the local spreading code of length 15 and  ***
;***   initialization for the acquisition-time measurement    ***
;*** For comments see the comments on generation of the local ***
;*** spreading code of length 7                               ***

Rec4            LRLK    AR3,PNData
                LARP    AR3
Rec4_0          LAC     *
                SUBK    4
                BZ      Rec4_1
                CALL    SKIPCODE
                B       Rec4_0
Rec4_1          RTC
                LAR     AR0,Len4
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10P
Rec4_2          CMPR    0
                BBNZ    Rec4_5,*+,AR3
                LAR     AR2,Taps
Rec4_3          LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    Rec4_3,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+,AR4
                LAC     *,15
                SFL
                ADD     NewSeq,12
                SACH    *
                LAC     *
                BNZ     Rec4_4
                LACK    -1
                SACL    *
Rec4_4          MAR     *+
                LARP    AR5
                LAC     NewSeq
                ANDK    0Fh
                SACL    Seq
                SFL
```

```
                SACL    NewSeq
                B       Rec4_2

Rec4_5          LRLK    AR2,15
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    15
                LARP    AR4
Rec4_6          LAC     *+,AR5
                SACL    *+,AR2
                BANZ    Rec4_6,*-,AR4

Rec4_7          LAR     AR0,Len4
                SAR     AR0,Len
                LALK    SSCode10P
                ADD     Phase
                SACL    SSCodeP
                LAR     AR1,SSCodeP
                LAR     AR2,ONE

                B       Corr_Init

;*** Generation of the local spreading code of length 31 and  ***
;***   initialization for the acquisition-time measurement    ***
;*** For comments see the comments on generation of the local ***
;*** spreading code of length 7                               ***

Rec5            LRLK    AR3,PNData
                LARP    AR3
Rec5_0          LAC     *
                SUBK    5
                BZ      Rec5_1
                CALL    SKIPCODE
                B       Rec5_0
Rec5_1          RTC
                LAR     AR0,Len5
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10P
Rec5_2          CMPR    0
                BBNZ    Rec5_5,*+,AR3
                LAR     AR2,Taps
Rec5_3          LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    Rec5_3,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+,AR4
                LAC     *,15
                SFL
                ADD     NewSeq,11
```

```
                SACH    *
                LAC     *
                BNZ     Rec5_4
                LACK    -1
                SACL    *
Rec5_4          MAR     *+
                LARP    AR5
                LAC     NewSeq
                ANDK    1Fh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       Rec5_2

Rec5_5          LRLK    AR2,31
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    31
                LARP    AR4
Rec5_6          LAC     *+,AR5
                SACL    *+,AR2
                BANZ    Rec5_6,*-,AR4

Rec5_7          LAR     AR0,Len5
                SAR     AR0,Len
                LALK    SSCode10P
                ADD     Phase
                SACL    SSCodeP
                LAR     AR1,SSCodeP
                LAR     AR2,ONE

                B       Corr_Init

;*** Generation of the local spreading code of length 63 and  ***
;***   initialization for the acquisition-time measurement    ***
;*** For comments see the comments on generation of the local ***
;*** spreading code of length 7                               ***

Rec6            LRLK    AR3,PNData
                LARP    AR3
Rec6_0          LAC     *
                SUBK    6
                BZ      Rec6_1
                CALL    SKIPCODE
                B       Rec6_0
Rec6_1          RTC
                LAR     AR0,Len6
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10P
Rec6_2          CMPR    0
                BBNZ    Rec6_5,*+,AR3
                LAR     AR2,Taps
Rec6_3          LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
```

```
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    Rec6_3,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+,AR4
                LAC     *,15
                SFL
                ADD     NewSeq,10
                SACH    *
                LAC     *
                BNZ     Rec6_4
                LACK    -1
                SACL    *
Rec6_4          MAR     *+
                LARP    AR5
                LAC     NewSeq
                ANDK    3Fh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       Rec6_2

Rec6_5          LRLK    AR2,63
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    63
                LARP    AR4
Rec6_6          LAC     *+,AR5
                SACL    *+,AR2
                BANZ    Rec6_6,*-,AR4

Rec6_7          LAR     AR0,Len6
                SAR     AR0,Len
                LALK    SSCode10P
                ADD     Phase
                SACL    SSCodeP
                LAR     AR1,SSCodeP
                LAR     AR2,ONE

                B       Corr_Init

;*** Generation of the local spreading code of length 127 and ***
;***   initialization for the acquisition-time measurement    ***
;*** For comments see the comments on generation of the local ***
;*** spreading code of length 7                               ***

Rec7            LRLK    AR3,PNData
                LARP    AR3
Rec7_0          LAC     *
                SUBK    7
                BZ      Rec7_1
                CALL    SKIPCODE
                B       Rec7_0
Rec7_1          RTC
                LAR     AR0,Len7
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10P
Rec7_2          CMPR    0
                BBNZ    Rec7_5,*+,AR3
```

```
                LAR     AR2,Taps
Rec7_3          LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    Rec7_3,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+,AR4
                LAC     *,15
                SFL
                ADD     NewSeq,9
                SACH    *
                LAC     *
                BNZ     Rec7_4
                LACK    -1
                SACL    *
Rec7_4          MAR     *+
                LARP    AR5
                LAC     NewSeq
                ANDK    7Fh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       Rec7_2

Rec7_5          LRLK    AR2,127
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    127
                LARP    AR4
Rec7_6          LAC     *+,AR5
                SACL    *+,AR2
                BANZ    Rec7_6,*-,AR4

Rec7_7          LAR     AR0,Len7
                SAR     AR0,Len
                LALK    SSCode10P
                ADD     Phase
                SACL    SSCodeP
                LAR     AR1,SSCodeP
                LAR     AR2,ONE

                B       Corr_Init

;*** Generation of the local spreading code of length 255 and ***
;***   initialization for the acquisition-time measurement    ***
;*** For comments see the comments on generation of the local ***
;*** spreading code of length 7                               ***

Rec8            LRLK    AR3,PNData
                LARP    AR3
Rec8_0          LAC     *
                SUBK    8
                BZ      Rec8_1
                CALL    SKIPCODE
                B       Rec8_0
Rec8_1          RTC
                LAR     AR0,Len8
```

```
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10P
Rec8_2          CMPR    0
                BBNZ    Rec8_5,*+,AR3
                LAR     AR2,Taps
Rec8_3          LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    Rec8_3,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+,AR4
                LAC     *,15
                SFL
                ADD     NewSeq,8
                SACH    *
                LAC     *
                BNZ     Rec8_4
                LACK    -1
                SACL    *
Rec8_4          MAR     *+
                LARP    AR5
                LAC     NewSeq
                ANDK    0FFh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       Rec8_2

Rec8_5          LRLK    AR2,255
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    255
                LARP    AR4
Rec8_6          LAC     *+,AR5
                SACL    *+,AR2
                BANZ    Rec8_6,*-,AR4

Rec8_7          LAR     AR0,Len8
                SAR     AR0,Len
                LALK    SSCode10P
                ADD     Phase
                SACL    SSCodeP
                LAR     AR1,SSCodeP
                LAR     AR2,ONE

                B       Corr_Init
```

```
;*** Generation of the local spreading code of length 511 and ***
;***   initialization for the acquisition-time measurement     ***
;*** For comments see the comments on generation of the local ***
;*** spreading code of length 7                               ***

Rec9            LRLK    AR3,PNData
                LARP    AR3
Rec9_0          LAC     *
                SUBK    9
                BZ      Rec9_1
                CALL    SKIPCODE
                B       Rec9_0
Rec9_1          RTC
                LAR     AR0,Len9
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10P
Rec9_2          CMPR    0
                BBNZ    Rec9_5,*+,AR3
                LAR     AR2,Taps
Rec9_3          LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    Rec9_3,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+,AR4
                LAC     *,15
                SFL
                ADD     NewSeq,7
                SACH    *
                LAC     *
                BNZ     Rec9_4
                LACK    -1
                SACL    *
Rec9_4          MAR     *+
                LARP    AR5
                LAC     NewSeq
                ANDK    1FFh
                SACL    Seq
                SFL
                SACL    NewSeq
                B       Rec9_2

Rec9_5          LRLK    AR2,511
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    255
                ADRK    255
                ADRK    1
                LARP    AR4
Rec9_6          LAC     *+,AR5
```

```
                SACL    *+,AR2
                BANZ    Rec9_6,*-,AR4

Rec9_7          LAR     AR0,Len9
                SAR     AR0,Len
                LALK    SSCode10P
                ADD     Phase
                SACL    SSCodeP
                LAR     AR1,SSCodeP
                LAR     AR2,ONE

                B       Corr_Init

;*** Generation of the local spreading code of length 1023 and ***
;***   initialization for the acquisition-time measurement     ***
;*** For comments see the comments on generation of the local  ***
;*** spreading code of length 7                                ***

Rec10           LRLK    AR3,PNData
                LARP    AR3
Rec10_0         LAC     *
                SUBK    10
                BZ      Rec10_1
                CALL    SKIPCODE
                B       Rec10_0
Rec10_1         RTC
                LAR     AR0,Len10
                MAR     *+
                LAR     AR2,*+
                SAR     AR2,Taps
                LAR     AR5,NIL
                LAC     *+,AR5
                SACL    Seq
                SFL
                SACL    NewSeq
                LRLK    AR4,SSCode10P
Rec10_2         CMPR    0
                BBNZ    Rec10_5,*+,AR3
                LAR     AR2,Taps
Rec10_3         LAR     AR1,*+,AR2
                SAR     AR1,Tap
                LACK    14
                SUB     Tap
                SACL    Tap
                LAC     Seq
                RPT     Tap
                SFL
                AND     Mask
                SACH    Tapx,1
                LAC     NewSeq
                XOR     Tapx
                SACL    NewSeq
                MAR     *-
                BANZ    Rec10_3,*,AR3
                RPT     Taps
                MAR     *-
                MAR     *+,AR4
                LAC     *,15
                SFL
                ADD     NewSeq,6
                SACH    *
                LAC     *
                BNZ     Rec10_4
                LACK    -1
                SACL    *
Rec10_4         MAR     *+
                LARP    AR5
                LAC     NewSeq
                ANDK    3FFh
                SACL    Seq
```

```
                SFL
                SACL    NewSeq
                B       Rec10_2

Rec10_5         LRLK    AR2,1023
                LRLK    AR4,SSCode10P
                LARP    AR5
                LRLK    AR5,SSCode10P
                ADRK    255
                ADRK    255
                ADRK    255
                ADRK    255
                ADRK    3
                LARP    AR4
Rec10_6         LAC     *+,AR5
                SACL    *+,AR2
                BANZ    Rec10_6,*-,AR4

Rec10_7         LAR     AR0,Len10
                SAR     AR0,Len
                LALK    SSCode10P
                ADD     Phase
                SACL    SSCodeP
                LAR     AR1,SSCodeP
                LAR     AR2,ONE

                B       Corr_Init

Corr_Init       ZAC                             ; initialize acquisition time
                SACL    Hi_Acq                  ; and correlation value
                SACL    Lo_Acq
                SACL    Hi_Corr
                SACL    Lo_Corr
                LRLK    AR3,SendData
                LARP    AR5

Enable_ADC_Int  LDPK    0
                LAC     IMR
                ORK     3
                SACL    IMR                     ; enable DAC- and ADC Interrupts
                LDPK    6                       ; DAC_Interrupt is used to count the clock
                                                ; ticks, to determine the acquisition time
                EINT
                B       MAINLOOP


;***********************************************************************
;
;       Stop command has been received, send Ok
;
;***********************************************************************


Stop            LRLK    AR0,OkStr               ; "Ok"
                CALL    Transmit
Stop_0          LAC     PAR_Status              ; Do nothing, give timer chance to
                BNZ     Stop_0                  ; generate last interrupt of "Ok"
                RPTK    255                     ; transmission
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
```

```
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h

                NOP

                DINT                            ; if a Stop command has been received
                RXF                             ; the transceiver is reset
                CNFD
                ROVM
                RSXM
                SPM     0
                LDPK    6

                B       Beg_L1


;***********************************************************************
;
;       Start command has been received, send Ok
;
;***********************************************************************

Start           LRLK    AR0,OkStr               ; "Ok"
                CALL    Transmit
Start_0         LAC     PAR_Status              ; Do nothing, give timer chance to
                BNZ     Start_0                 ; generate last interrupt of "Ok"
                RPTK    255                     ; transmission
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h

                NOP

                DINT

                IN      InData,1                ; reset flip flop U20

                LAC     Device
                BNZ     Start_1

;********** transmitter has received the start command ******

                B       Data_L3                 ; start generation of spreading codes again

;**********  receiver has received the start command ******

Start_1         LDPK    0
                LAC     IMR
                ORK     2
```

```
                SACL    IMR                     ; enable ADC_Interrupt
                LDPK    6
                EINT

                B       MAINLOOP

;***********************************************************************
;
;       Reset command has been received, send Ok and restart program
;
;***********************************************************************

Reset           LRLK    AR0,OkStr               ; "Ok"
                CALL    Transmit
Reset_0         LAC     PAR_Status              ; Do nothing, give timer chance to
                BNZ     Reset_0                 ; generate last interrupt of "Ok"
                RPTK    255                     ; transmission
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h

                B       Begin                   ; branch to main program start

;*****************************************************************
;
;  DAC_Interrupt
;
;  This interrupt procedure calculates the signal amplitude
;  of the modulated data signal and the spreaded random data signals
;  of a certain number of other users the calculated signal amplitude
;  is scaled and written to the DAC
;
;*****************************************************************

DAC_Int         DINT

                LARP    AR7                     ; save environment
                MAR     *-
                SST1    *-                      ; save status register ST1
                SST     *-                      ; save status register ST0
                SACH    *-                      ; save high accumulator
                SACL    *-,AR5                  ; save low accumulator

                LAC     Device
                BNZ     DAC_5

                BANZ    DAC_1,*,AR1
                LAC     Users
                SACL    Count                   ; reset all counters and pointers because
                LAR     AR1,ONE                 ; end of message has been reached
                LAR     AR2,SSCodeM             ; transmission of the message is repeated
                LRLK    AR3,SSUser16
                SAR     AR3,AR3_16
                LRLK    AR3,SSUser15
```

```
                SAR     AR3,AR3_15
                LRLK    AR3,SSUser14
                SAR     AR3,AR3_14
                LRLK    AR3,SSUser13
                SAR     AR3,AR3_13
                LRLK    AR3,SSUser12
                SAR     AR3,AR3_12
                LRLK    AR3,SSUser11
                SAR     AR3,AR3_11
                LRLK    AR3,SSUser10
                SAR     AR3,AR3_10
                LRLK    AR3,SSUser9
                SAR     AR3,AR3_9
                LRLK    AR3,SSUser8
                SAR     AR3,AR3_8
                LRLK    AR3,SSUser7
                SAR     AR3,AR3_7
                LRLK    AR3,SSUser6
                SAR     AR3,AR3_6
                LRLK    AR3,SSUser5
                SAR     AR3,AR3_5
                LRLK    AR3,SSUser4
                SAR     AR3,AR3_4
                LRLK    AR3,SSUser3
                SAR     AR3,AR3_3
                LRLK    AR3,SSUser2
                SAR     AR3,AR3_2
                LRLK    AR3,SSUser1
                LRLK    AR4,SendData
                LAR     AR5,SendLen
DAC_1           CMPR    0                       ; has the end of the code-length
                BBNZ    DAC_2,*+,AR3            ; period been reached?
                LAC     Count
                BZ      DAC_1U0
                SUBK    1
                SACL    Count
                LAC     RandomSeq,1             ; load first bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+,AR2                  ; determine the chip of user 1
                ADD     *+,AR3
                SACL    OutData
                SAR     AR3,AR3_1
                LAR     AR3,AR3_2
                LAC     Count
                BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
                SUBK    1
                SACL    Count
                LAC     RandomSeq,2             ; load second bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+                      ; determine the chip of user 2
                ADD     OutData
                SACL    OutData
                SAR     AR3,AR3_2
                LAR     AR3,AR3_3
                LAC     Count
                BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
                SUBK    1
                SACL    Count
                LAC     RandomSeq,3             ; load third bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+                      ; determine the chip of user 3
                ADD     OutData
                SACL    OutData
                SAR     AR3,AR3_3
```

```
LAR     AR3,AR3_4
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,4             ; load fourth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 4
ADD     OutData
SACL    OutData
SAR     AR3,AR3_4
LAR     AR3,AR3_5
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,5             ; load fifth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 5
ADD     OutData
SACL    OutData
SAR     AR3,AR3_5
LAR     AR3,AR3_6
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,6             ; load sixth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 6
ADD     OutData
SACL    OutData
SAR     AR3,AR3_6
LAR     AR3,AR3_7
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,7             ; load seventh bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 7
ADD     OutData
SACL    OutData
SAR     AR3,AR3_7
LAR     AR3,AR3_8
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,8             ; load eighth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 8
ADD     OutData
SACL    OutData
SAR     AR3,AR3_8
LAR     AR3,AR3_9
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
```

```
LAC     RandomSeq,9             ; load ninth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 9
ADD     OutData
SACL    OutData
SAR     AR3,AR3_9
LAR     AR3,AR3_10
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,10            ; load tenth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 10
ADD     OutData
SACL    OutData
SAR     AR3,AR3_10
LAR     AR3,AR3_11
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,11            ; load eleventh bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 11
ADD     OutData
SACL    OutData
SAR     AR3,AR3_11
LAR     AR3,AR3_12
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,12            ; load twelvth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 12
ADD     OutData
SACL    OutData
SAR     AR3,AR3_12
LAR     AR3,AR3_13
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,13            ; load thirteenth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 13
ADD     OutData
SACL    OutData
SAR     AR3,AR3_13
LAR     AR3,AR3_14
LAC     Count
BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,14            ; load fourteenth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 14
```

```
                ADD     OutData
                SACL    OutData
                SAR     AR3,AR3_14
                LAR     AR3,AR3_15
                LAC     Count
                BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
                SUBK    1
                SACL    Count
                LAC     RandomSeq,15            ; load fifteenth bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+                      ; determine the chip of user 15
                ADD     OutData
                SACL    OutData
                SAR     AR3,AR3_15
                LAR     AR3,AR3_16
                LAC     Count
                BZ      DAC_FINISH              ; if no more users than branch to DAC_FINISH
                SUBK    1
                SACL    Count
                ZALH    RandomSeq               ; load sixteenth bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+                      ; determine the chip of user 16
                ADD     OutData
                SACL    OutData
                SAR     AR3,AR3_16
                B       DAC_FINISH
DAC_1U0         LARP    AR2                     ; executed when no other users
                LAC     *+
                SACL    OutData
DAC_FINISH      LAR     AR3,AR3_1
                LARP    AR6
                LAC     OutData                 ; scale output value
                ADD     Amplitude
                SACL    Amp
                LAR     AR6,Amp
                LAC     *,AR4
                SACL    OutData
                OUT     OutData,2               ; generate DAC-output signal

                LAC     Users
                SACL    Count
                LARP    AR7                     ; restore environment
                MAR     *+
                ZALS    *+                      ; load low accumulator
                ADDH    *+                      ; load high accumulator
                LST     *+                      ; load status register ST0
                LST1    *+,AR5                  ; load status register ST1

                EINT
                RET
DAC_2           RTC
                LARP    AR4
                MAR     *+,AR3
                LAC     Count
                BZ      DAC_2U0
                SUBK    1
                SACL    Count
                LAC     RandomSeq,1             ; load first bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+,AR2                  ; determine the chip of user 1
                ADD     *+,AR3
                SACL    OutData
                SAR     AR3,AR3_1
                LAR     AR3,AR3_2
```

```
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,2             ; load second bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 2
ADD     OutData
SACL    OutData
SAR     AR3,AR3_2
LAR     AR3,AR3_3
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,3             ; load third bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 3
ADD     OutData
SACL    OutData
SAR     AR3,AR3_3
LAR     AR3,AR3_4
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,4             ; load fourth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 4
ADD     OutData
SACL    OutData
SAR     AR3,AR3_4
LAR     AR3,AR3_5
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,5             ; load fifth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 5
ADD     OutData
SACL    OutData
SAR     AR3,AR3_5
LAR     AR3,AR3_6
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,6             ; load sixth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 6
ADD     OutData
SACL    OutData
SAR     AR3,AR3_6
LAR     AR3,AR3_7
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,7             ; load seventh bit of shift register
```

```
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 7
ADD     OutData
SACL    OutData
SAR     AR3,AR3_7
LAR     AR3,AR3_8
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,8             ; load eighth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 8
ADD     OutData
SACL    OutData
SAR     AR3,AR3_8
LAR     AR3,AR3_9
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,9             ; load ninth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 9
ADD     OutData
SACL    OutData
SAR     AR3,AR3_9
LAR     AR3,AR3_10
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,10            ; load tenth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 10
ADD     OutData
SACL    OutData
SAR     AR3,AR3_10
LAR     AR3,AR3_11
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,11            ; load eleventh bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 11
ADD     OutData
SACL    OutData
SAR     AR3,AR3_11
LAR     AR3,AR3_12
LAC     Count
BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
SUBK    1
SACL    Count
LAC     RandomSeq,12            ; load twelvth bit of shift register
ANDK    8000h,1                 ; that generates the random signal
SACH    Bit
LAC     Bit
XOR     *+                      ; determine the chip of user 12
ADD     OutData
```

```
                SACL    OutData
                SAR     AR3,AR3_12
                LAR     AR3,AR3_13
                LAC     Count
                BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
                SUBK    1
                SACL    Count
                LAC     RandomSeq,13            ; load thirteenth bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+                      ; determine the chip of user 13
                ADD     OutData
                SACL    OutData
                SAR     AR3,AR3_13
                LAR     AR3,AR3_14
                LAC     Count
                BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
                SUBK    1
                SACL    Count
                LAC     RandomSeq,14            ; load fourteenth bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+                      ; determine the chip of user 14
                ADD     OutData
                SACL    OutData
                SAR     AR3,AR3_14
                LAR     AR3,AR3_15
                LAC     Count
                BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
                SUBK    1
                SACL    Count
                LAC     RandomSeq,15            ; load fifteenth bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+                      ; determine the chip of user 15
                ADD     OutData
                SACL    OutData
                SAR     AR3,AR3_15
                LAR     AR3,AR3_16
                LAC     Count
                BZ      DAC_2FINISH             ; if no more users than branch to DAC_FINISH
                SUBK    1
                SACL    Count
                ZALH    RandomSeq               ; load sixteenth bit of shift register
                ANDK    8000h,1                 ; that generates the random signal
                SACH    Bit
                LAC     Bit
                XOR     *+                      ; determine the chip of user 16
                ADD     OutData
                SACL    OutData
                SAR     AR3,AR3_16
                B       DAC_2U0
DAC_2U0         LARP    AR2                     ; executed when no other users are present
                LAC     *+
                SACL    OutData
DAC_2FINISH     LAR     AR3,AR3_1
                LARP    AR6
                LAC     OutData                 ; scale output value
                ADD     Amplitude
                SACL    Amp
                LAR     AR6,Amp
                LAC     *,AR4
                SACL    OutData
                OUT     OutData,2               ; generate DAC_output signal
                LAC     *,AR5
                BZ      DAC_3                   ; reset all counters and pointers
                LAR     AR2,SSCodeP             ; because the end of the code-length
```

```
                B       DAC_4                   ; period has been reached
DAC_3           LAR     AR2,SSCodeM
DAC_4           LRLK    AR3,SSUser16
                SAR     AR3,AR3_16
                LRLK    AR3,SSUser15
                SAR     AR3,AR3_15
                LRLK    AR3,SSUser14
                SAR     AR3,AR3_14
                LRLK    AR3,SSUser13
                SAR     AR3,AR3_13
                LRLK    AR3,SSUser12
                SAR     AR3,AR3_12
                LRLK    AR3,SSUser11
                SAR     AR3,AR3_11
                LRLK    AR3,SSUser10
                SAR     AR3,AR3_10
                LRLK    AR3,SSUser9
                SAR     AR3,AR3_9
                LRLK    AR3,SSUser8
                SAR     AR3,AR3_8
                LRLK    AR3,SSUser7
                SAR     AR3,AR3_7
                LRLK    AR3,SSUser6
                SAR     AR3,AR3_6
                LRLK    AR3,SSUser5
                SAR     AR3,AR3_5
                LRLK    AR3,SSUser4
                SAR     AR3,AR3_4
                LRLK    AR3,SSUser3
                SAR     AR3,AR3_3
                LRLK    AR3,SSUser2
                SAR     AR3,AR3_2
                LRLK    AR3,SSUser1
                LAC     Users
                SACL    Count
                LAR     AR1,ONE

                LAC     RandomSeq               ; generate new random signal using
                AND     Mask1                   ; a 16-bit feedback shift register
                SACL    Spare
                ADD     Spare,4
                ADD     Spare,13
                ADD     Spare,15
                AND     Mask1
                ADDH    RandomSeq
                SACH    RandomSeq,1

                MAR     *-

                LARP    AR7                     ; restore environment
                MAR     *+
                ZALS    *+                      ; load low accumulator
                ADDH    *+                      ; load high accumulator
                LST     *+                      ; load status register ST0
                LST1    *+,AR5                  ; load status register ST1

                EINT
                RET

DAC_5           ZALS    Lo_Time                 ; this part of the interrupt routine is
                ADDH    Hi_Time                 ; executed when the transceiver is configured
                ADD     ONE                     ; as receiver. The number of transmit-clock
                SACL    Lo_Time                 ; ticks is counted to determine the
                SACH    Hi_Time                 ; acquisition time

                LAC     Hi_Time
                SUB     ReqTime
                BLZ     DAC_6                   ; if the maximum required time
                                                ; exceeded, the receiver stops acquiring
```

```
                LDPK    0
                LAC     IMR
                ANDK    0FFFCh
                SACL    IMR                     ; disable the DAC- and ADC-interrupts
                LDPK    6

                LACK    1
                SACL    Above                   ; Above is monitored in the main-program loop

DAC_6           LARP    AR7                     ; restore environment
                MAR     *+
                ZALS    *+                      ; load low accumulator
                ADDH    *+                      ; load high accumulator
                LST     *+                      ; load status register ST0
                LST1    *+,AR5                  ; load status register ST1

                EINT
                RET

;***************************************************************************
;
;  X_Int and R_Int are not used
;
;***************************************************************************

X_Int           RET

R_Int           RET

;***************************************************************************
;
;  The correct code phase has been acquired and data
;  were despreaded. Now the measurement data are stored
;  in 4-bit words to transmit them to the personal computer
;
;***************************************************************************

Despread_End    LAC     DataLen
                SFR
                SFR
                SFR
                SACL    DataLen             ; DataLen contains length of received message
                LARP    AR4                 ; in bytes
                LAR     AR4,DataLen
                SBRK    1
                LRLK    AR3,SendData
                LRLK    AR5,SendBufData
                LARP    AR3
Despread_1      ZAC                         ; set despreaded data in 4-bit words
                LAC     *+,3
                ADD     *+,2
                ADD     *+,1
                ADD     *+,AR5
                SACL    *+,AR3
                LAC     *+,3
                ADD     *+,2
                ADD     *+,1
                ADD     *+,AR5
                SACL    *+,AR4
                BANZ    Despread_1,*-,AR3

                LARP    AR0                 ; make a string to be transmitted to the personal
                LRLK    AR0,PAR_TrmData     ; computer. The string contains 4-bit words
                LACK    0                   ; starting with 'Data ', or in hexadecimal ASCII
                SACL    *+                  ; 4, 4, 6, 1, 7, 4, 6, 1. These data are preceeded
                LAC     DataLen,1           ; by the SOH and the length of the packet.
                SACL    DataLen
                ADDK    24
                SACL    DataLen
                SACL    PAR_TrmLen
```

```
                LAC     DataLen,8
                ANDK    0F000h,4
                SACH    *+
                LAC     DataLen,12
                ANDK    0F000h,4
                SACH    *+
                ZALH    DataLen
                ANDK    0F000h,4
                SACH    *+
                LACK    4
                SACL    *+
                SACL    *+
                LACK    6
                SACL    *+
                LACK    1
                SACL    *+
                LACK    7
                SACL    *+
                LACK    4
                SACL    *+
                LACK    6
                SACL    *+
                LACK    1
                SACL    *+
                LACK    2
                SACL    *+
                ZAC
                SACL    *+
                LAC     Hi_Acq,4                ; 'Data ' is followed by
                SACH    *+                      ; the high word of the acquisition time,
                LAC     Hi_Acq,8                ; the low word of the acquisition time,
                ANDK    0F000h,4                ; the number of verifications and the
                SACH    *+                      ; despreaded data (all in 4-bit words).
                LAC     Hi_Acq,12
                ANDK    0F000h,4
                SACH    *+
                ZALH    Hi_Acq
                ANDK    0F000h,4
                SACH    *+
                LAC     Lo_Acq,4
                SACH    *+
                LAC     Lo_Acq,8
                ANDK    0F000h,4
                SACH    *+
                LAC     Lo_Acq,12
                ANDK    0F000h,4
                SACH    *+
                ZALH    Lo_Acq
                ANDK    0F000h,4
                SACH    *+
                LAC     Verification,12
                SACH    *+
                ZALH    Verification
                ANDK    0F000h,4
                SACH    *+,AR5

                LRLK    AR5,SendBufData
                LAC     DataLen
                SUBK    25
                SACL    DataLen
                LAR     AR4,DataLen
Despread_2      LAC     *+,AR0
                SACL    *+,AR4
                BANZ    Despread_2,*-,AR5

                ZAC
                SACL    EndOfADC_Int            ; reset EndOfADC_Int

                LRLK    AR1,PAR_TrmData
                LACK    3
```

```
                SACL    PAR_Status              ; set PAR_Status to 3
                ZAC
                SACL    PAR_Retry               ; reset PAR_Retry
                SACL    PAR_ZeroCnt             ; reset PAR_ZeroCnt

                LAC     ONE
                SACL    PAR_TrmCnt              ; set PAR_TrmCnt to 1
                LptWrtC SOH                     ; start transmitting the first
                                                ; character....

Despread_3      LAC     PAR_Status
                BNZ     Despread_3              ; Do nothing, give timer chance to
                RPTK    255                     ; generate last interrupt data
                LAC     300h                    ; transmission
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h

                NOP

                DINT
                RXF
                CNFD
                ROVM
                RSXM
                SPM     0
                LDPK    6

                B       Beg_L1                  ; reset program after Data has been sent

;*************************************************************************
;
;  The maximum required time to acquire the correct code phase
;  has been exceeded. Acquisition is stopped and the message
;  "T_acq above required time" is sent to the personal computer.
;
;*************************************************************************

TooHigh         LRLK    AR0,TooHighStr          ; "T_acq above required time"
                CALL    Transmit
TooHigh_0       LAC     PAR_Status
                BNZ     TooHigh_0               ; Do nothing, give timer chance to
                RPTK    255                     ; generate last interrupt of data
                LAC     300h                    ; transmission
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
```

```
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h
                RPTK    255
                LAC     300h

                NOP

                DINT
                RXF
                CNFD
                ROVM
                RSXM
                SPM     0
                LDPK    6

                B       Beg_L1                  ; processor is reset after string
                                                ; has been transmitted to the PC

;***********************************************************************
;
;  Data for DAC scaling
;
;***********************************************************************

Amp0    .word  0FD0h, 0030h
Amp1    .word  0FD0h, 0800h, 0030h
Amp2    .word  0FD0h, 0C9Eh, 0362h, 0030h
Amp3    .word  0FD0h, 0DA8h, 0800h, 0258h, 0030h
Amp4    .word  0FD0h, 0E32h, 0B94h, 046Ch, 01CEh, 0030h
Amp5    .word  0FD0h, 0E88h, 0C9Eh, 0800h, 0362h, 0178h, 0030h
Amp6    .word  0FD0h, 0EC3h, 0D3Dh, 0B06h, 04FAh, 02C3h, 013Dh, 0030h
Amp7    .word  0FD0h, 0EEEh, 0DA8h, 0C00h, 0800h, 0400h, 0258h, 0112h, 0030h
Amp8    .word  0FD0h, 0F0Dh, 0DF6h, 0C9Eh, 0AABh, 0555h, 0362h, 0209h, 00F2h
        .word  0030h
Amp9    .word  0FD0h, 0F28h, 0E32h, 0D0Fh, 0B94h, 0800h, 046Ch, 02F1h, 01CDh
        .word  00D8h, 0030h
Amp10   .word  0FD0h, 0F3Ch, 0E62h, 0D65h, 0C2Eh, 0A69h, 0597h, 03D2h, 029Bh
        .word  019Eh, 00C4h, 0030h
Amp11   .word  0FD0h, 0F4Eh, 0E88h, 0DA8h, 0CB1h, 0B44h, 0800h, 04BCh, 034Fh
        .word  0258h, 0178h, 00B2h, 0030h
Amp12   .word  0FD0h, 0F5Bh, 0EA8h, 0DDFh, 0CF6h, 0BD8h, 0A38h, 05C8h, 0428h
        .word  030Ah, 0221h, 0158h, 00A4h, 0030h
Amp13   .word  0FD0h, 0F68h, 0EC3h, 0E0Ch, 0D3Dh, 0C47h, 0B06h, 0800h, 04FAh
        .word  03B9h, 02C3h, 01F4h, 013Dh, 0098h, 0030h
Amp14   .word  0FD0h, 0F73h, 0EDAh, 0E32h, 0D77h, 0C9Eh, 0B94h, 0A11h, 05EFh
        .word  046Ch, 0362h, 0289h, 01CEh, 0126h, 008Dh, 0030h
Amp15   .word  0FD0h, 0F7Ch, 0EEEh, 0E53h, 0DA8h, 0CE8h, 0C00h, 0AD4h, 0800h
        .word  052Ch, 0400h, 031Ah, 0258h, 01ADh, 0112h, 0084h, 0030h
Amp16   .word  0FD0h, 0F84h, 0EFFh, 0E6Fh, 0DD2h, 0D22h, 0C57h, 0B5Ch, 09F1h
        .word  060Fh, 04A4h, 03A9h, 02DEh, 022Eh, 0191h, 0101h, 007Ch, 0030h

;***********************************************************************
;
;  Data to generate own PN sequences
;
;  shift-register length, number of feedback taps, initial contents
;  of the shift register, feedback-tap positions.
;
;***********************************************************************

                .label SRDataBg
SRData3         .word   3, 2,         111b, 2, 0
SRData4         .word   4, 2,        1111b, 3, 0
SRData5         .word   5, 2,       11111b, 4, 1
SRData6         .word   6, 2,      111111b, 5, 0
SRData7         .word   7, 2,     1111111b, 6, 2
SRData8         .word   8, 4,    11111111b, 7, 3, 2, 1
```

```
SRData9         .word   9, 2,  111111111b, 8, 3
SRData10        .word  10, 2, 1111111111b, 9, 2
                .label SRDataEnd
SRDataLen       .set SRDataEnd-SRDataBg-1

;***************************************************************************
;
;  Data to generate PN sequences of other users
;
;***************************************************************************

                .label SRUserDataBg
SRUserData5     .word   5, 4,      00001b, 4, 3, 2, 1
                .word   5, 4,      00101b, 4, 3, 1, 0
SRUserData6     .word   6, 4,     001001b, 5, 4, 1, 0
                .word   6, 4,     001101b, 5, 4, 3, 1
SRUserData7     .word   7, 2,    0100001b, 6, 0
                .word   7, 4,    0110001b, 6, 2, 1, 0
                .word   7, 4,    0001101b, 6, 3, 2, 1
                .word   7, 4,    0000101b, 6, 5, 3, 1
                .word   7, 4,    0001001b, 6, 5, 2, 0
                .word   7, 4,    0111001b, 6, 5, 4, 1
                .word   7, 6,    1001101b, 6, 5, 4, 3, 1, 0
                .word   7, 6,    1110001b, 6, 4, 3, 2, 1, 0
                .word   7, 2,    1001101b, 6, 0
                .word   7, 4,    1101011b, 6, 2, 1, 0
                .word   7, 4,    1010101b, 6, 3, 2, 1
                .word   7, 4,    0001011b, 6, 5, 3, 1
                .word   7, 4,    0100101b, 6, 5, 2, 0
                .word   7, 4,    1010011b, 6, 5, 4, 1
                .word   7, 6,    0101001b, 6, 5, 4, 3, 1, 0
                .word   7, 6,    1000001b, 6, 4, 3, 2, 1, 0
;SRUserData8     .word   8, 4,   10000001b, 7, 5, 4, 2
;                .word   8, 4,   00111001b, 7, 4, 2, 0
;                .word   8, 4,   10001011b, 7, 5, 4, 1
;                .word   8, 4,   01101001b, 7, 5, 4, 0
;                .word   8, 4,   00110001b, 7, 6, 5, 0
;                .word   8, 6,   00001111b, 7, 6, 5, 4, 1, 0
;                .word   8, 6,   11010001b, 7, 6, 3, 2, 1, 0
;SRUserData9     .word   9, 4,  010110001b, 8, 5, 3, 2
;                .word   9, 4,  010001001b, 8, 7, 4, 3
;                .word   9, 4,  111000001b, 8, 7, 3, 0
;                .word   9, 4,  001111101b, 8, 4, 2, 1
;                .word   9, 4,  001111111b, 8, 7, 5, 4
;                .word   9, 4,  010101011b, 8, 7, 6, 1
;                .word   9, 6,  001010101b, 8, 5, 4, 3, 1, 0
;                .word   9, 6,  101000001b, 8, 6, 5, 3, 2, 0
;SRUserData10    .word  10, 4, 0100101111b, 9, 7, 2, 1
;                .word  10, 6, 1001011101b, 9, 5, 4, 2, 1, 0
;                .word  10, 6, 0110101001b, 9, 8, 7, 5, 2, 1
;                .word  10, 8, 0011111001b, 9, 8, 7, 6, 5, 4, 3, 2
;                .word  10, 8, 0010111101b, 9, 7, 6, 5, 4, 3, 2, 0
                .label SRUserDataEnd
SRUserLen       .set SRUserDataEnd-SRUserDataBg-1

;***************************************************************************
;
;  Memory space to store calculated PN codes and related data
;
;***************************************************************************

Ampl0           .usect "DataRAM",2
Ampl1           .usect "DataRAM",3
Ampl2           .usect "DataRAM",4
Ampl3           .usect "DataRAM",5
Ampl4           .usect "DataRAM",6
Ampl5           .usect "DataRAM",7
Ampl6           .usect "DataRAM",8
Ampl7           .usect "DataRAM",9
Ampl8           .usect "DataRAM",10
```

```
Ampl9           .usect "DataRAM",11
Ampl10          .usect "DataRAM",12
Ampl11          .usect "DataRAM",13
Ampl12          .usect "DataRAM",14
Ampl13          .usect "DataRAM",15
Ampl14          .usect "DataRAM",16
Ampl15          .usect "DataRAM",17
Ampl16          .usect "DataRAM",18
SendBufData     .usect "DataRAM",64
SendData        .usect "DataRAM",256
PNData          .usect "DataRAM",64
PNUserData      .usect "DataRAM",256
SSCode10P       .usect "DataRAM",2048
SSCode10M       .usect "DataRAM",2048
SSUser1         .usect "UserCode",1024
SSUser2         .usect "UserCode",1024
SSUser3         .usect "UserCode",1024
SSUser4         .usect "UserCode",1024
SSUser5         .usect "UserCode",1024
SSUser6         .usect "UserCode",1024
SSUser7         .usect "UserCode",1024
SSUser8         .usect "UserCode",1024
SSUser9         .usect "UserCode",1024
SSUser10        .usect "UserCode",1024
SSUser11        .usect "UserCode",1024
SSUser12        .usect "UserCode",1024
SSUser13        .usect "UserCode",1024
SSUser14        .usect "UserCode",1024
SSUser15        .usect "UserCode",1024
SSUser16        .usect "UserCode",1024
abs_end

                .asect "ProgB0",0FF00h
                .label PrgBgBlk0

;***********************************************************************
;
;  ADC_Interrupt
;
;  This interrupt routine calculates the correlation of the input
;  signal and the local spreading code. If this correlation exceeds
;  the threshold level for three successive period, acquisition is
;  declared and the received signal is despread. In addition, the
;  received data are sent to the personal computer.
;
;***********************************************************************

ADC_Int         DINT

                LARP    AR7                     ; save environment
                MAR     *-
                SST1    *-                      ; save status register ST1
                SST     *-                      ; save status register ST0
                SACH    *-                      ; save high accumulator
                SACL    *-,AR1                  ; save low accumulator

                SSXM

                IN      InData,2
                NOP
                LAC     InData,2
                SACL    InData

                ZALS    Lo_Corr
                ADDH    Hi_Corr
                LT      *+,AR2
                MPY     InData
                APAC
                SACL    Lo_Corr
                SACH    Hi_Corr
```

```
                CMPR    0                       ; end of dwell-time?
                BBNZ    ADC_1,*+,AR1

                RSXM
                                                ; restore environment
                LARP    AR7
                MAR     *+                      ; load low accumulator
                ZALS    *+                      ; load high accumulator
                ADDH    *+                      ; load status register ST0
                LST     *+                      ; load status register ST1
                LST1    *+,AR5

                EINT
                RET

ADC_1           LAC     Acq
                BNZ     ADC_8
                LT      Lo_Th
                MPYK    1
                LPH     Hi_Th
                ZALS    Lo_Corr
                ADDH    Hi_Corr
                ABS
                SPAC
                BGEZ    ADC_3                   ; |correlation| above Threshold?

                ZAC
                SACL    VerNr
                MAR     *0-
                LAR     AR2,ONE

                ZAC
                LT      *+,AR2
                MPY     InData
                APAC
                SACL    Lo_Corr
                SACH    Hi_Corr

                MAR     *+,AR7

                RSXM
                                                ; restore environment
                MAR     *+
                ZALS    *+                      ; load low accumulator
                ADDH    *+                      ; load high accumulator
                LST     *+                      ; load status register ST0
                LST1    *+,AR5                  ; load status register ST1

                EINT
                RET

ADC_2           MAR     *0-,AR7                 ; if not then look for next phase
                LAR     AR2,ONE
                ZAC
                SACL    Lo_Corr
                SACL    Hi_Corr

                RSXM
                                                ; restore environment
                MAR     *+
                ZALS    *+                      ; load low accumulator
                ADDH    *+                      ; load high accumulator
                LST     *+                      ; load status register ST0
                LST1    *+,AR5                  ; load status register ST1

                EINT
                RET

ADC_3           LAC     VerNr
                BNZ     ADC_4
```

```
                LAC     Verification
                ADDK    1
                SACL    Verification
                ZAC
ADC_4           ADDK    1
                SACL    VerNr
                SUB     VerifyNr
                BZ      ADC_5                   ; if correlation is 3 times
                                                ; above threshold then
                B       ADC_2                   ; acquisition occurred and
                                                ; data can be decoded

ADC_5           LACK    1                       ; acquisition is declared
                SACL    Acq

                LDPK    0                       ; stop counting acquisition time
                LAC     IMR
                ANDK    0FFFEh
                SACL    IMR
                LDPK    6

                ZALH    Hi_Corr
                BLZ     ADC_6,*,AR1
                LACK    1
                SACL    OneCnt
                B       ADC_2
ADC_6           B       ADC_2

ADC_8           LAC     Frame_Sync
                BNZ     ADC_12
                ZALH    Hi_Corr
                BLZ     ADC_9,*,AR1
                LAC     OneCnt
                ADDK    1
                SACL    OneCnt
                SUBK    7
                BZ      ADC_11                  ; if 7 successive ones are received
                B       ADC_2                   ; then bit synchronization is ok
ADC_9           ZAC
                SACL    OneCnt
                B       ADC_2

ADC_11          LACK    1                       ; bit synchronization is ok
                SACL    Frame_Sync
                SAR     AR3,Frame_Bg
                B       ADC_2

ADC_12          LAC     Data_Sync               ; look if bit is 1 or 0
                BNZ     ADC_17                  ; and store in memory
                ZALH    Hi_Corr
                BLZ     ADC_13,*,AR3
                LACK    1
                SACL    *+,AR1
                B       ADC_14
ADC_13          ZAC
                SACL    *+,AR1
ADC_14          LAC     LenCnt                  ; get length of message in bytes
                ADDK    1                       ; these are stored in the next 8 bits
                SACL    LenCnt                  ; to receive
                SUBK    8
                BZ      ADC_15
                B       ADC_2

ADC_15          LACK    1
                SACL    Data_Sync               ; length has been received
                LRLK    AR3,SendData
                LRLK    AR4,SendData
                LARP    AR4
                ZAC
                LAC     *+,7
```

```
                ADD     *+,6
                ADD     *+,5
                ADD     *+,4
                ADD     *+,3
                ADD     *+,2
                ADD     *+,1
                ADD     *+,AR1
                SACL    DataLen                 ; store length of message to receive
                SUBK    31
                BLZ     ADC_16
                LACK    31
                SACL    DataLen
ADC_16          LAC     DataLen,3
                SACL    DataLen
                LAR     AR4,DataLen
                B       ADC_2

ADC_17          ZALH    Hi_Corr                 ; stores received bits in memory
                BLZ     ADC_18,*,AR3
                LACK    1
                SACL    *+,AR4
                B       ADC_19
ADC_18          ZAC
                SACL    *+,AR4
ADC_19          BANZ    ADC_2,*-,AR1

                LDPK    0                       ; disable interrupts 0 and 1
                LAC     IMR                     ; if data is despreaded
                ANDK    0FFFCh
                SACL    IMR
                LDPK    6

                LAC     Hi_Time
                SACL    Hi_Acq
                LAC     Lo_Time
                SACL    Lo_Acq

                LALK    0800h
                SACL    OutData
                OUT     OutData,2

                LACK    1
                SACL    EndOfADC_Int

                B       ADC_2

;***************************************************************************
;
;  Procedure to clear flip flop U20
;  This procedure must be executed from the internal processor RAM
;  to avoid data-bus conflicts.
;
;***************************************************************************

ClrADC          IN InData,2
                NOP
                RET

                .label PrgEndBlk0
PrgLenBlk0      .set PrgEndBlk0 - PrgBgBlk0 - 1

;***************************************************************************
;
;  Program end
;
;***************************************************************************
                .end
```

Appendix 5: PC program source

```
program acquisition;          {This program provides communication via parallel}
                              {port with the baseband transmitter and receiver }
                              {to control the measurement procedure.           }
                              {Acquisition-time measurements are performed     }
                              {fully automatically. The program structure is   }
                              {described in Chapter 7 of the graduation report.}
                              {The communications software is described in     }
                              {Chapter 6, which discusses the communications   }
                              {interface.                                      }


uses dos,crt;                 {Note: interrupt procedures that handle          }
                              {      hardware-generated interrupts should not  }
                              {      use any of Turbo Pascal's input and output}
                              {      or dynamic memory allocation routines,    }
                              {      because they are not reentrant. Likewise  }
                              {      no DOS functions can be used because DOS  }
                              {      is not reentrant. See Turbo Pascal Ref.   }

const
       lptnr=1;               {parallel port number                            }
       timeoutlen=2;          {(-1 no timeout) or time length = n*1/18 sec (n>1)}
       timeoutlenansw=10;     {(-1 no timeout) or time length = n*1/18 sec (n>1)}
       timeint=$08;           {the clock tick interrupt                        }
       lpt1int=$0F;           {the parallel port 1 interrupt                   }
       lpt2int=$0D;           {the parallel port 2 interrupt                   }
       aack=$0C;              {attention acknowledge 1100                      }
       dack=$0A;              {data acknowledge      1010                      }
       eot=$06;               {end of transmission   0110                      }
       nack=$03;              {not acknowledge       0011                      }
       soh=$00;               {start of header       0000                      }


type
      bytearray=array[0..255] of byte;
      wordarray=array[0..255] of word;
      commdata=record
                 status:integer;  {0=inactive, 1=transmit, 2=receive, 3=answer}
                 retry:integer;           {number of retries made  \  4=error}
                 transmlen:integer;       {length of transmit message       }
                 transmcount:integer;     {counter for characters to transmit}
                 transmdata:bytearray;    {characters to trasnmit           }
                 receivlen:integer;       {length of received message       }
                 receivcount:integer;     {counter for characters received  }
                 receivdata:bytearray;    {characters received              }
               end; {record commdata}


var
     address:word;
     value:byte;
     c:longint;
     nr:integer;                      {exitcode                             }
     packet:commdata;
     data:byte;                       {databyte read from parallel port     }
     lpt:integer;                     {address of parallel port             }
     tocount:integer;                 {timeout counter                      }
     transmitarray:bytearray;         {array to be transmitted              }
     messagearray:bytearray;          {array to be printed to screen        }
```

```
    oldtimerint,oldlptint:pointer;     {pointers to the standard interrupts   }
    commnr:integer;                    {device number, to be set by dip-sws   }
    outdata0,outdata1:byte;            {bytes for device selection            }
                                       {and interrupt generation              }
    writevalue:byte;                   {byte to be send to parallel port      }
    q,x,y:integer;
    count:integer;
    temp:byte;
    len:byte;                          {length of character string to send    }
    str:string;                        {string to send                        }
    parameters:string;                 {string indicating where parameters    }
                                       {are stored                            }
    toohigh_hi,toohigh_lo:byte;        {high and low byte of maximum          }
                                       {acquisition time                      }
    acq_data:text;                     {file to write measurement data to     }
    param:text;                        {file to read parameters from          }
    filename:string;                   {actual name of file to write data to  }
    recmessage,message:string;         {received message, data to modulate    }
    srlen,users:byte;                  {shift-register length, number of users}
    verification:byte;                 {number of verifications to perform    }
    ver_number:byte;                   {number of verification modes entered  }
    phase,pnlen:word;                  {random code phase, PN code length     }
    toohigh:word;                      {maximum acquisition time              }
    time_hi,time_lo:longint;           {high and low byte of acquisition time }
    th_hi,th_lo:longint;               {high and low byte of threshold        }
    threshold:longint;                 {threshold level                       }
    acq_time:real;                     {acquisition time                      }
    lower,upper:integer;               {lower and upper threshold level       }


procedure restorelptint;               {this procedure restores the standard  }
                                       {hardware printer interrupt vector     }
begin
  if (lpt=$3BC) or (lpt=$378) then     {printer port is LPT1}
  begin
    port[$21]:=port[$21] or $80;       {disable interrupt via interrupt mask  }
    setintvec(lpt1int,oldlptint);      {reset the interrupt vector            }
  end
  else
  begin
    if lpt=$278 then                   {printer port is LPT2}
    begin
      port[$21]:=port[$21] or $20;     {disable interrupt via interrupt mask  }
      setintvec(lpt2int,oldlptint);    {reset the interrupt vector            }
    end;
  end;
  port[lpt+2]:=4;
end;  {restorelptint}



procedure restoretimerint;             {this procedure restores the standard }
                                       {clock-tick interrupt vector          }
begin
  setintvec(timeint,oldtimerint);
end;  {restoretimerint}



procedure quit(nr:integer);

begin            {exits the program with exit code: 0 if communication ended}
  restorelptint;                                    { 1 if lpt? not found     }
  restoretimerint;
  halt(nr);
end;  {quit}
```

```
procedure lptwrite(writevalue:byte);     {this procedure writes a value       }
                                         {(writevalue) to the printer port and}
                                         {generates an interrupt on the device}

begin
  port[lpt]:=writevalue;                 {write data to printer port           }
  port[lpt+2]:=outdata1;                 {select device and make IRQ on device low}
  tocount:=timeoutlen;
  if writevalue<16 then
  begin
    memw[$B000:q+480]:=writevalue+$0E30;  {write the value that is sent to screen}
    memw[$B000:q+481]:=$02;               {writevalue is control signal        }
  end
  else
  begin
    memw[$B000:q+480]:=writevalue+$0E00;  {write the value that is sent to screen}
    memw[$B000:q+481]:=$02;               {writevalue is character             }
  end;
  q:=q+2;
  if q>480 then q:=0;                     {maximum of 3 lines of values sent   }
                                          {written to screen                   }
  port[lpt+2]:=outdata0;                  {make IRQ on device high             }
  port[lpt+2]:=outdata1;                  {make IRQ on device low              }
end;  {lptwrite}



procedure incretry_t;                    {another attempt to transmit is made}

begin
  packet.retry:=packet.retry+1;          {count the number of retries         }
  if packet.retry=3 then packet.status:=4  {if number of retries equals 3 than }
                                           {fatal error, communication aborted }
  else
  begin
    packet.transmcount:=1;
    packet.transmdata[0]:=$01;
    lptwrite(soh);                        {else send start of header (SOH) again }
                                          {and initialise parameters           }
  end;
end;  {incretry_t}



procedure incretry_r;                    {another attempt to receive is made.  }
                                         {This happens if AACK or EOT not received}
                                         {when expected, or timeout on receive }
                                         {occurred                             }
begin
  packet.retry:=packet.retry+1;          {count the number of retries         }
  if packet.retry=3 then packet.status:=4  {if number of retries equals 3 than }
                                           {fatal error, communication aborted }
  else
  begin
    lptwrite(nack);                      {else send not acknowledge (NACK)    }
    packet.receivcount:=0;               {and initialise parameters           }
  end;
end;  {incretry_r}



procedure maketextarray(charstring:string; var text:bytearray);

var  counter:integer;                    {this procedure makes a textarray of a  }
     ch:byte;                            {string to be send. The textarray contains}
     letter:char;                        {the ordinal numbers of the string characters}
     longintvar:longint;
     str:string;
     code:integer;
```

```
begin
  len:=length(charstring);
  for counter:=1 to len do
  begin
    letter:=charstring[counter];
    ch:=ord(letter);
    text[counter-1]:=ch;
  end;
  text[len]:=13;                          {terminate textarray with value 13       }
end;  {maketextarray}



procedure lptmsg(var message:bytearray);

var  a:integer;                           {this procedure sends a message to the   }
     teller:integer;                      {screen and scrolls the old messages     }
     temp:word;                           {upwards. The message is not written to  }
                                          {the screen with the 'write' command     }
                                          {because within a hardware-interrupt     }
                                          {routine it is not allowed to use DOS    }
                                          {commands. The message is directly       }
                                          {written to the video memory. $B000 is   }
                                          {the base address of the Hercules video  }
                                          {memory, $B800 of the VGA video memory   }
begin
  for a:=1840 to 1919 do
  begin
    memw[$B000:a shl 1]:=memw[$B000:(a shl 1)+160];  {scroll old message upwards }
    memw[$B000:(a shl 1)+1]:=$02;
  end;
  a:=1920;
  teller:=0;
  while message[teller]<>13 do            {message is terminated with value 13     }
  begin
    memw[$B000:a shl 1]:=message[teller]+$0E00;  {write character to video memory}
    memw[$B000:(a shl 1)+1]:=$02;                {determine the character color  }
    a:=a+1;
    teller:=teller+1;
  end;
  while a<2000 do
  begin
    memw[$B000:a shl 1]:=$0E20;           {fill rest of line with spaces           }
    memw[$B000:(a shl 1)+1]:=$02;
    a:=a+1;
  end;
end; {lptmsg}



{$F+,S-}
procedure timerint;
interrupt;

begin                                     {this procedure replaces the standard}
  asm                                     {clock tick interrupt                 }
    cli;                                  {disable all interrupts               }
  end;
  if tocount<>-1 then                     {if tocount=-1 then no timeout check }
  begin
    tocount:=tocount-1;
    if tocount=0 then
    begin
      if packet.status=1 then             {timeout occurred in transmit mode   }
      begin
        port[lpt+2]:=outdata1;            {make IRQ on device low              }
        maketextarray('Error: Timeout on transmit',messagearray);
        lptmsg(messagearray);             {write the above message to screen   }
        incretry_t;                       {retry transmission                  }
      end
```

```
      else
      begin
        if packet.status=2 then          {timeout occurred in receive mode    }
        begin
          port[lpt+2]:=outdata1;         {make IRQ on device low              }
          maketextarray('Error: Timeout on receive',messagearray);
          lptmsg(messagearray);          {write the above message to screen   }
          incretry_r;                    {retry receive                       }
        end
        else                             {unexpected timeout occurred         }
        begin
          maketextarray('Error: unexpected Timeout',messagearray);
          lptmsg(messagearray);          {write the above message to screen   }
        end;
      end;
    end;
  end;
  asm
    sti;                                 {enable all interrupts               }
    call oldtimerint;                    {the standard clock-tick interrupt is}
  end;                                   {also executed to ensure normal      }
                                         {PC operation                        }
  port[$20]:=$20;       {send end-of-interrupt signal to interrupt controller}
end;  {timerint}
{$F-,S+}



procedure inittimerint;                  {this procedure sets the vector of   }
                                         {the clock-tick interrupt to the     }
                                         {'timerint' procedure and stores the}
                                         {old interrupt vector in oldtimerint}
begin
  getintvec(timeint,oldtimerint);
  setintvec(timeint,@timerint);
end;  {inittimerint}



procedure lpttransmit(transmitarray:bytearray);

var  x:integer;         {this procedure makes the text array to be transmitted}
                        {including header and data, and starts communication  }
                        {by sending the first character (SOH)                 }

begin
  outdata0:=(((commnr shl 1) or $10) xor 4) xor $0E;
  outdata1:=outdata0 or 1;
  if (lpt=$378) or (lpt=$3bc) then port[$21]:=port[$21] or $80
  else port[$21]:=port[$21] or $20;      {disable printer interrupt via imask }
  port[lpt+2]:=outdata1;
  delay(15);
  if (lpt=$378) or (lpt=$3bc) then port[$21]:=port[$21] and $7F
  else port[$21]:=port[$21] or $DF;      {enable printer interrupt via imask  }
  packet.status:=1;                      {initialize variables to start       }
  packet.transmlen:=len+2;               {communication with device           }
  packet.transmdata[0]:=soh;
  packet.transmdata[1]:=packet.transmlen;
  x:=0;
  while x<len do
  begin
    packet.transmdata[2+x]:=transmitarray[x];
    x:=x+1;
  end;
  packet.receivcount:=0;
  packet.retry:=0;
  packet.transmcount:=1;
  lptwrite(soh);                         {start transmission of SOH}
end;  {lpttransmit}
```

```
{$F+,S-}
procedure lptint;
interrupt;

var x:integer;                          {replaces the printer interrupt routine.}
                                        {The communications protocol described  }
                                        {in Chapter 6 of the graduation report  }
                                        {is implemeted in the interrupt routine.}
begin
  asm
    cli;                                {disable all interrupts                 }
  end;
  tocount:=-1;
  data:=port[lpt+1];                    {read data from Centronics port         }
  data:=((data and $38) shr 3)+(((data and $80) xor $80) shr 4);
  memw[$B000:q]:=data+$0E30;            {write data received to screen          }
  memw[$B000:q+1]:=$02;                 {determine the character color          }
  q:=q+2;
  if q>480 then q:=0;                   {maximum of 3 lines is written to screen}
  if (packet.status=0) and (data=SOH) then  {measurement data have been         }
  begin                                     {received from baseband receiver    }
    packet.status:=2;
    packet.receivcount:=0;
    packet.retry:=0;
  end;
  if packet.status=1 then               {transmit mode}
  begin
    if packet.transmcount=1 then
    begin
      if data<>aack then
      begin
        packet.transmdata[0]:=packet.transmdata[0]+1;
        if packet.transmdata[0]>$09 then
        begin
          incretry_t;
          maketextarray('Error: AACK not received on ATTENTION',messagearray);
          lptmsg(messagearray);
        end
        else lptwrite(soh);
      end
      else
      begin
        lptwrite(packet.transmdata[packet.transmcount]);
        packet.transmcount:=packet.transmcount+1;
      end;
    end
    else
    begin
      if packet.transmcount<packet.transmlen then
      begin
        if data<>dack then
        begin
          maketextarray('Error: other code than DACK received on DATA',messagearray);
          lptmsg(messagearray);
          incretry_t;
        end
        else
        begin
          lptwrite(packet.transmdata[packet.transmcount]);
          packet.transmcount:=packet.transmcount+1;
        end;
      end
      else
      begin
        if data<>eot then
        begin
          maketextarray('Error: EOT not received on last DATA',messagearray);
          lptmsg(messagearray);
          incretry_t;
```

```
      end
      else
      begin
        lptwrite(eot);
        packet.status:=2;              {after transmission, go into receive mode    }
        tocount:=timeoutlenansw;       {allow more time before answer is transmitted}
      end;
    end;
  end;
end
else
begin
  if packet.status=2 then              {receive mode}
  begin
    if packet.receivcount=0 then
    begin
      if data<>soh then
      begin
        incretry_r;
        maketextarray('Error: first reception not correct',messagearray);
        lptmsg(messagearray);
      end
      else
      begin
        lptwrite(aack);
        packet.receivcount:=packet.receivcount+1;
      end;
    end
    else
    begin
      if packet.receivcount=1 then
      begin
        lptwrite(dack);
        packet.receivlen:=data shl 8;
        packet.receivcount:=packet.receivcount+1;
      end
      else
      begin
        if packet.receivcount=2 then
        begin
          lptwrite(dack);
          packet.receivlen:=packet.receivlen+(data shl 4);
          packet.receivcount:=packet.receivcount+1;
        end
        else
        begin
          if packet.receivcount=3 then
          begin
            lptwrite(dack);
            packet.receivlen:=packet.receivlen+data;
            packet.receivcount:=packet.receivcount+1;
          end
          else
          if packet.receivcount+1<packet.receivlen then
          begin
            lptwrite(dack);
            packet.receivdata[packet.receivcount-4]:=data;
            packet.receivcount:=packet.receivcount+1;
          end
          else
          begin
            if packet.receivcount+1=packet.receivlen then
            begin
              lptwrite(eot);
              packet.receivdata[packet.receivcount-4]:=data;
              packet.receivcount:=packet.receivcount+1;
            end
            else
            begin
              if data<>eot then
```

```
                begin
                  maketextarray('Error: last EOT not correctly received',messagearray);
                  lptmsg(messagearray);
                  incretry_r;
                end
                else
                begin
                  for x:=0 to ((packet.receivlen-4) shr 1)-1 do
                  begin
                    packet.receivdata[x]:=(packet.receivdata[x shl 1] shl 4)
                                          +packet.receivdata[1+(x shl 1)];
                  end;
                  packet.receivdata[x+1]:=13;
                  packet.status:=3;   {after message is received, go into answer mode}
                end;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
  if lpt=$278 then port[$20]:=$65
  else port[$20]:=$67;
  asm
    sti;                                {enable all interrupts}
  end;
end;  {lptint}
{$F-,S+}



function initlptint:integer;            {this procedure sets the hardware   }
                                        {interrupt vector to the new defined}
                                        {procedure 'lptint' and stores the  }
                                        {old vector in oldlptint            }

begin
  lpt:=0;
  packet.status:=0;
  lpt:=memw[$0000:$406+lptnr*2];        {read port address     }
  if lpt=0 then initlptint:=0           {no port has been found}
  else
  begin
    port[lpt]:=255;
    if (lpt=$378) or (lpt=$3BC) then    {port is LPT1          }
    begin
      getintvec(lpt1int,oldlptint);
      setintvec(lpt1int,@lptint);
      port[$21]:=port[$21] and $7F;     {enable printer interrupt via imask}
    end
    else
    begin
      if lpt=$278 then                  {port is LPT2          }
      begin
        getintvec(lpt2int,oldlptint);
        setintvec(lpt2int,@lptint);
        port[$21]:=port[$21] and $DF;   {enable printer interrupt via imask}
      end;
    end;
    port[lpt+2]:=((commnr shl 1) or $10) xor 4;  {enable interrupt via     }
    initlptint:=1;                               {pin 10 of Centronics port}
  end;
end;  {initlptint}
```

```
procedure measure;                          {this procedure sends measurement  }
                                            {parameters to the transmitter and }
                                            {to the receiver and receives the  }
                                            {measurement data from the receiver}
begin
  q:=0;
  randomize;
  packet.status:=0;
  tocount:=-1;
  commnr:=0;                                {select transmitter}
  len:=length(message);
  phase:=random(pnlen);                     {generate random phase between own }
                                            {spreading code and codes of other users}
  str:='Data '+chr(srlen)+chr(phase mod 256)+chr(phase div 256);
  str:=str+chr(users)+chr(127)+chr(len)+message+'.';
  maketextarray(str,messagearray);          {data sent to transmitter concern: }
                                            {- shift register length           }
                                            {- low byte of phase               }
                                            {- high byte of phase              }
                                            {- number of other users           }
                                            {- prefix code of message          }
                                            {- message to be modulated and     }
                                            {  and transmitted                 }
  lpttransmit(messagearray);                {start transmission}
  repeat                                    {wait until data have been sent to}
    if packet.status=4 then quit(1)         {transmitter and answer has been  }
  until packet.status=3;                    {received                         }
  gotoxy(10,10);
  writeln('Data was sent to transmitter.');
  gotoxy(10,11);
  write('Answer: ');
  c:=0;
  while packet.receivdata[c]<>13 do
  begin
    write(chr(packet.receivdata[c]));       {write answer to screen}
    c:=c+1;
  end;
  writeln;
  q:=0;
  delay(random(pnlen));                     {generate random phase delay between}
  packet.status:=0;                         {transmitter and receiver          }
  tocount:=-1;
  commnr:=1;
  toohigh_hi:=toohigh div 256;
  toohigh_lo:=toohigh mod 256;
  str:='Data '+chr(srlen)+chr(th_hi div 256)+chr(th_hi mod 256);
  str:=str+chr(th_lo div 256)+chr(th_lo mod 256)+chr(verification);
  str:=str+chr(toohigh_hi)+chr(toohigh_lo)+chr(phase mod 256)+chr(phase div 256);
  maketextarray(str,messagearray);          {data send to receiver concern:   }
                                            {- shift register length          }
                                            {- threshold (32 bit word sent    }
                                            {  in 4 bytes)                    }
                                            {- number of successive           }
                                            {  verifications to perform       }
                                            {- maximum acquisition time       }
                                            {- phase that indicates first bit }
                                            {  of the spreading code          }
  lpttransmit(messagearray);                {start transmission               }
  repeat                                    {wait until data have been sent to}
    if packet.status=4 then quit(1)         {receiver and answer has been     }
  until packet.status=3;                    {received                         }
  packet.status:=0;
  q:=0;
  gotoxy(10,12);
  writeln('Data was sent to receiver.');
  gotoxy(10,13);
  write('Answer: Data received. Despreading...');
  c:=0;
  while packet.receivdata[c]<>13 do
  begin
```

```
  write(chr(packet.receivdata[c]));   {write answer to screen}
  c:=c+1;
end;
writeln;
gotoxy(10,14);
writeln('Measuring...');
repeat                                {wait until measurement data have }
  if packet.status=4 then quit(1)     {been received                    }
until packet.status=3;
recmessage:='';
for c:=1 to 25 do recmessage[c]:=chr(packet.receivdata[c]);
if recmessage<>'T_acq above required time' then
begin                                 {if receiver indicates that maximum}
                                      {acquisition time was exceeded, do }
                                      {nothing and start next measurement}
  recmessage:='';
  time_hi:=(packet.receivdata[5] shl 8)+packet.receivdata[6];
  time_lo:=(packet.receivdata[7] shl 8)+packet.receivdata[8];
  acq_time:=(time_hi*65536+time_lo)/16000;   {clock frequency of DAC_Int is 16 kHz}
  ver_number:=packet.receivdata[9];          {for details on DAC_Int, see assembly listing}
  c:=10;
  while packet.receivdata[c]<>13 do
  begin
    recmessage:=recmessage+chr(packet.receivdata[c]);
    c:=c+1;
  end;
  assign(acq_data,filename);          {write measurement data to file}
  append(acq_data);
  writeln(acq_data,acq_time:1:8,'  ',ver_number:3,'  ',recmessage);
  close(acq_data);
  gotoxy(10,15);
  writeln('Data despreaded.');        {write measurement data to screen}
  gotoxy(10,16);
  writeln('Number of other users: ',users);
  gotoxy(10,17);
  writeln('Threshold: ',y,'  X: ',x,'   ');
  gotoxy(10,18);
  writeln('T_acq = ',acq_time:3:5,' sec      ');
  gotoxy(10,19);
  writeln('Number of verifications = ',ver_number,'    ');
  gotoxy(10,20);
  write('Data received: ');
  c:=10;
  while packet.receivdata[c]<>13 do
  begin
    write(chr(packet.receivdata[c]));
    c:=c+1;
  end;
  writeln('                          ');
end
else
begin
  assign(acq_data,filename);
  append(acq_data);
  writeln(acq_data,'');
  close(acq_data);
  gotoxy(10,15);
  writeln('Data not despreaded within required time.');
end;
q:=0;
packet.status:=0;
tocount:=-1;
commnr:=0;
str:='Stop';
maketextarray(str,messagearray);
lpttransmit(messagearray);            {send stop command to transmitter   }
repeat                                {wait until command has been sent to}
  if packet.status=4 then quit(1)     {transmitter and answer has been    }
until packet.status=3;                {received                           }
gotoxy(10,21);
```

```
  writeln('Stop command was sent to transmitter.');
  gotoxy(10,22);
  write('Answer: ');
  c:=0;
  while packet.receivdata[c]<>13 do
  begin
    write(chr(packet.receivdata[c]));   {write answer to screen}
    c:=c+1;
  end;
  writeln;
  delay(4);
end;



procedure measurements;                 {this procedure initializes a        }
                                        {measurement procedure by resetting  }
                                        {the transmitter and the receiver,   }
                                        {and reads the measuement parameters }
                                        {from file. It opens the output file }
                                        {and resets the transmitter and      }
                                        {receiver again when the measurements}
                                        {were performed                      }
begin
  delay(500);
  q:=0;
  clrscr;
  packet.status:=0;
  tocount:=-1;
  commnr:=1;                            {select receiver                     }
  str:='Reset';
  maketextarray(str,messagearray);
  lpttransmit(messagearray);            {send reset command to receiver      }
  repeat                                {wait until command has been sent to}
    if packet.status=4 then quit(1)     {receiver and answer has been        }
  until packet.status=3;                {received                            }
  gotoxy(10,8);
  writeln('Reset command was sent to receiver.');
  gotoxy(10,9);
  write('Answer: ');
  c:=0;
  while packet.receivdata[c]<>13 do
  begin
    write(chr(packet.receivdata[c]));   {write answer to screen}
    c:=c+1;
  end;
  writeln;
  packet.status:=0;
  tocount:=-1;
  commnr:=0;                            {select transmitter                  }
  str:='Reset';
  maketextarray(str,messagearray);
  lpttransmit(messagearray);            {send reset command to transmitter   }
  repeat                                {wait until command has been sent to}
    if packet.status=4 then quit(1)     {transmitter and answer has been     }
  until packet.status=3;                {received                            }
  gotoxy(10,10);
  writeln('Reset command was sent to transmitter.');
  gotoxy(10,11);
  write('Answer: ');
  c:=0;
  while packet.receivdata[c]<>13 do
  begin
    write(chr(packet.receivdata[c]));   {write answer to screen}
    c:=c+1;
  end;
  writeln;
  delay(1500);
  pnlen:=1;
  clrscr;
```

```
assign(param,parameters);          {read measurement parameters from file        }
reset(param);
readln(param,filename);            {output filename                              }
readln(param,srlen);               {length of shift register                     }
readln(param,users);               {number of other users                        }
readln(param,message);             {message to be modulated by transmitter       }
readln(param,threshold);           {threshold to which correlation is compared   }
readln(param,verification);        {number of successive verifications to perform}
readln(param,toohigh);             {time after which receiver stops acquiring    }
readln(param,lower);               {lower threshold                              }
readln(param,upper);               {upper threshold                              }
close(param);
for x:=1 to srlen do pnlen:=pnlen*2;
pnlen:=pnlen-1;                    {calculate length of spreading code           }
assign(acq_data,filename);         {open output file and write measurement       }
rewrite(acq_data);                 {parameters to this file                      }
writeln(acq_data,'Shift register length: ',srlen);
writeln(acq_data,'Number of other users: ',users);
writeln(acq_data,'Message: ',message);
writeln(acq_data,'Threshold: ',threshold);
writeln(acq_data,'Verification period: ',verification);
writeln(acq_data,'Maximum acquisition time: ',4.096*toohigh);
writeln(acq_data,'Lower threshold: ',lower);
writeln(acq_data,'Upper threshold: ',upper);
close(acq_data);
for y:=lower to upper do             {perform for threshold levels from lower to }
begin                                {upper 100 acquisition-time measurements    }
  th_hi:=y*threshold div 65536;
  th_lo:=y*threshold mod 65536;
  for x:=1 to 100 do
  begin
    measure;
  end;
  assign(acq_data,filename);         {add blank line to output file after each   }
  append(acq_data);                  {100 measurements                           }
  writeln(acq_data,'');
  close(acq_data);
end;
packet.status:=0;
q:=0;
clrscr;
tocount:=-1;
commnr:=0;                           {select transmitter                 }
str:='Reset';
maketextarray(str,messagearray);
lpttransmit(messagearray);           {send reset command to transmitter  }
repeat                               {wait until command has been sent to}
  if packet.status=4 then quit(1)    {transmitter and answer has been    }
until packet.status=3;               {received                           }
gotoxy(10,8);
writeln('Reset command was sent to transmitter.');
gotoxy(10,9);
write('Answer: ');
c:=0;
while packet.receivdata[c]<>13 do
begin
  write(chr(packet.receivdata[c]));  {write answer to screen             }
  c:=c+1;
end;
writeln;
packet.status:=0;
tocount:=-1;
commnr:=1;                           {select receiver                    }
str:='Reset';
maketextarray(str,messagearray);
lpttransmit(messagearray);           {send reset command to receiver     }
repeat                               {wait until command has been sent to}
  if packet.status=4 then quit(1)    {receiver and answer has been       }
until packet.status=3;               {received                           }
gotoxy(10,10);
```

```
  writeln('Reset command was sent to receiver.');
  gotoxy(10,11);
  write('Answer: ');
  c:=0;
  while packet.receivdata[c]<>13 do
  begin
    write(chr(packet.receivdata[c]));   {write answer to screen}
    c:=c+1;
  end;
  writeln;
  delay(1500);
end;


begin  {program}
  clrscr;
  inittimerint;
  if initlptint<>0 then
  begin
    parameters:='data71.prm';           {parameter file is 'data71.prm          }
    measurements;                       {perform measurements with 1 other user  }
    parameters:='data72.prm';           {parameter file is 'data72.prm          }
    measurements;                       {perform measurements with 2 other users }
    parameters:='data73.prm';           {parameter file is 'data73.prm          }
    measurements;                       {perform measurements with 3 other users }
    parameters:='data74.prm';           {parameter file is 'data74.prm          }
    measurements;                       {perform measurements with 4 other users }
    parameters:='data75.prm';           {parameter file is 'data75.prm          }
    measurements;                       {perform measurements with 5 other users }
    parameters:='data76.prm';           {parameter file is 'data76.prm          }
    measurements;                       {perform measurements with 6 other users }
    parameters:='data77.prm';           {parameter file is 'data77.prm          }
    measurements;                       {perform measurements with 7 other users }
    parameters:='data78.prm';           {parameter file is 'data78.prm          }
    measurements;                       {perform measurements with 8 other users }
    parameters:='data79.prm';           {parameter file is 'data79.prm          }
    measurements;                       {perform measurements with 9 other users }
    parameters:='data710.prm';          {parameter file is 'data710.prm         }
    measurements;                       {perform measurements with 10 other users}
    parameters:='data711.prm';          {parameter file is 'data711.prm         }
    measurements;                       {perform measurements with 11 other users}
    parameters:='data712.prm';          {parameter file is 'data712.prm         }
    measurements;                       {perform measurements with 12 other users}
    parameters:='data713.prm';          {parameter file is 'data713.prm         }
    measurements;                       {perform measurements with 13 other users}
    parameters:='data714.prm';          {parameter file is 'data714.prm         }
    measurements;                       {perform measurements with 14 other users}
    parameters:='data715.prm';          {parameter file is 'data715.prm         }
    measurements;                       {perform measurements with 15 other users}
  end
  else
  begin
    writeln('Error: LPT ',lptnr,' not found');
    quit(1);
  end;
end.  {program}
```

The **parameter file** contains following information:

```
d:\work\jos\commint\7_1.dat        {output filename}
7                                  {shift-register length}
1                                  {number of other users}
Joske                              {message to modulate and to transmit}
$00005A80                          {threshold level of the acquisition system}
3                                  {number of successive verifications to perform}
$0010                              {maximum acquisition time, $1 equals 4.096 seconds at}
                                   {16 kHz clock frequency}
20                                 {lower threshold compared to maximum correlation (=127)}
92                                 {upper threshold compared to maximum correlation (=127)}
```