APPLICATION NOTE

Equivalent Circuit Simulation for Inductively Coupled Transponders

AN99021

Philips
Semiconductors

PHILIPS

Equivalent Circuit Simulation for Inductively Coupled Transponders

Application Note AN99021

Abstract

The Philips product family PCF79xx offers devices for contact- and battery less identification applications, based on inductive coupling. This document derives a mathematical description of the operating principles and provides a simulation software package to assist the system designer in

- optimization

- worst case analysis

- evaluation and verification

of a system incorporating these devices.

APPLICATION NOTE

Equivalent Circuit Simulation for Inductively Coupled Transponders

AN99021

Author(s):

Wolfgang Tobergte
Frank Böh

Philips System Laboratory
Hamburg, Germany

Keywords

PCF79xx, Contactless identification, Transponder, Inductive Coupling, MATLAB®, Simulation

Date: 6. April 1999

Equivalent Circuit Simulation for Inductively Coupled Transponders

Application Note AN99021

Summary

This report is intended to provide application support for designing identification systems with the PCF79xx transponder family. It contains the necessary theory for design and optimization of the most critical pathes of the contactless interface; a concrete simulation program is presented.

The first section gives an overview of the operating principles of a contactless identification system employing the PCF79xx transponder family. Section 2 derives mathematical models for each aspect of the inductive link to be designed: the energy transmission to the transponder, data modulation from transponder to basestation, and demodulation principles at basestation side. The implementation of the derived equations in a computer simulation with the tools MATLAB® is presented in section 3. When verifying the simulation against an actual design, the measurement methods presented in section 4 will be useful.

Equivalent Circuit Simulation for Inductively Coupled Transponders

Application Note AN99021

CONTENTS

1. System overview

The transponders described in this document consist of an antenna configured as a resonant circuit, an EEPROM, and modulator and demodulator circuits for data transmission. The PCF79735 incorporates additional cryptographic security features. The basestation comprises to a second resonant circuit antenna, a driver stage powering the antenna, modulator and demodulator circuits, and usually a micro controller which handles the communication protocol with the transponder. Data are transmitted in either direction across the inductive link.

Figure 1 shows the general c**onf**iguration of an inductively coupled identification system using Philips transponders. An exemplary setup for the basestation part of the system is shown, together with a PCF79xx, transponder model.

The inductive link can be identified in the middle, consisting of two coils that together form a loosely coupled transformer. Two capacitors, one at basestation and a second at transponder side, form resonant circuits with each of the coils.

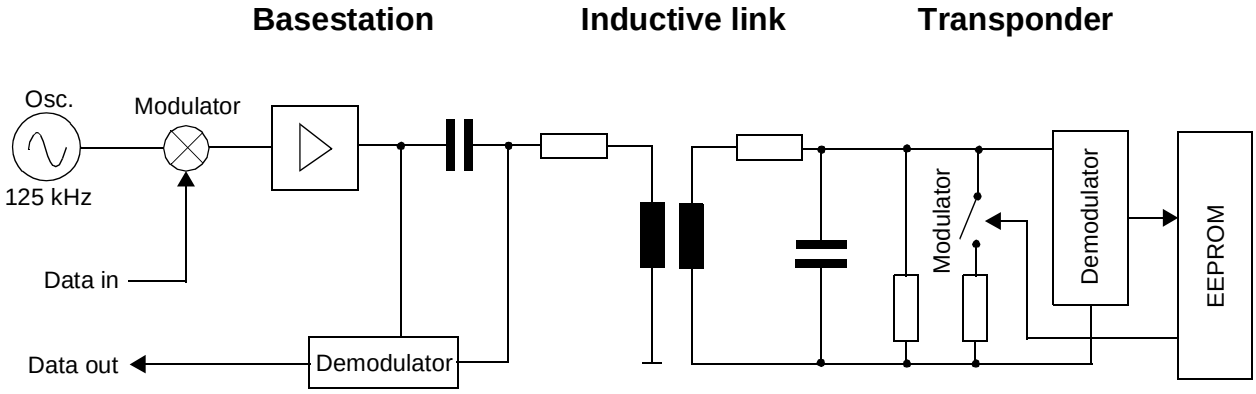


Fig.1 System overview

The transponder derives its power supply directly from the alternating magnetic field that is permanently generated by the basestation antenna coil. The power consumption of the transponder has direct influence to the system design. A mathematical description is developed in section 2.3.

Basestation and transponder contain each a modulator and a demodulator, which allows for data transmissions in both directions.

In the so-called Read-Mode, or “Get” direction, the data in the EEPROM of the transponder is transmitted to the basestation. The power consumption of the transponder is modelled by an equivalent impedance. A second impedance is switched by the transponder according to the binary data to be sent. This so-called load modulation is reflected back to the basestation side, which will be described in detail in section 2.4.

According to the transformer principle, the modulation can be seen at the basestation side by the varying impedance of the antenna coil, which itself causes a modulated (complex) antenna current. The most practical way of measuring this current is to measure the voltage across the resonant capacitor as shown in figure 1, since that provides an inherent HF noise rejection. This voltage is demodulated to restore the original binary information; different approaches will be discussed in section 2.5.

The Program-Mode, or “Put” direction, allows the basestation to program data into the transponder´s EEPROM. The basestation performs an amplitude modulation of the magnetic field, typically by switching on and off the power drivers according to the data stream. The varying magnetic field will induce a varying voltage in the transponder coil. The transponder´s demodulator will finally recover the binary information. The Program-Mode behaviour is mainly determined by transient processes, whereas the simulation models of this document are derived in the frequency domain. However, it will be covered with respect to power consumption in section 2.3.

2. Simulation models

In order to simulate the energy and data transmission performance of the system shown in figure 1, a set of equivalent circuits is derived in sections 2.1 and 2.2. The transformer model, in particular its electrical equivalent circuit, is used to describe the inductive link between basestation and transponder. Resonant frequency deviations, due to component tolerances of the two resonant circuits, are having the greatest impact to the system performance. The simulation models presented here are mainly based on varying these parameters.

Note: The simulation employs typical parameters. For a worst case analysis, all combination of the minimum or maximum values have to be considered.

2.1 Inductive link from basestation to transponder

The circuit diagram in figure 2 shows the relevant components that form together the inductive link of the system in figure 1. The two antenna coils L_1 (basestation) and L_2 (transponder) form a loosely coupled transformer with a coupling coefficient k. U_1 denotes the driver output voltage, whereas U_2 is the open circuit voltage at the transponder antenna with the transponder IC module removed. R_1 and R_2 denote the resistive part of the corresponding resonant circuit; C_1 and C_2 are the resonant capacitors at antenna and transponder side, respectively. Note that R_2 is the cumulative resistive part of L_2 and C_2, whereas R_1 additionally includes the output resistance of the basestation driver stage (see also section 4.2). The resistive parts of C_1 and C_2 can generally be neglected for R_1, R_2.

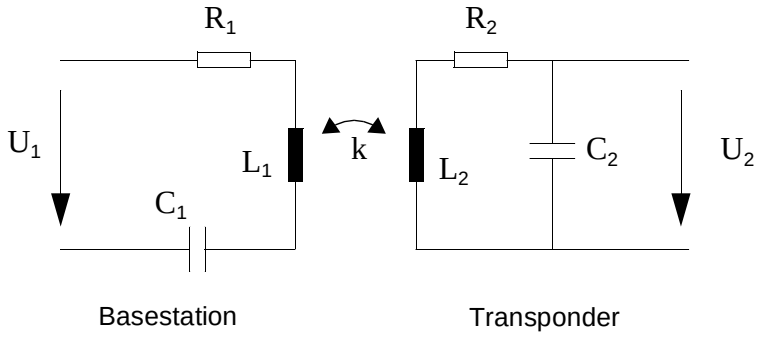


Fig.2 Inductive link

The transformer is replaced by its equivalent circuit in order to derive a mathematical description:

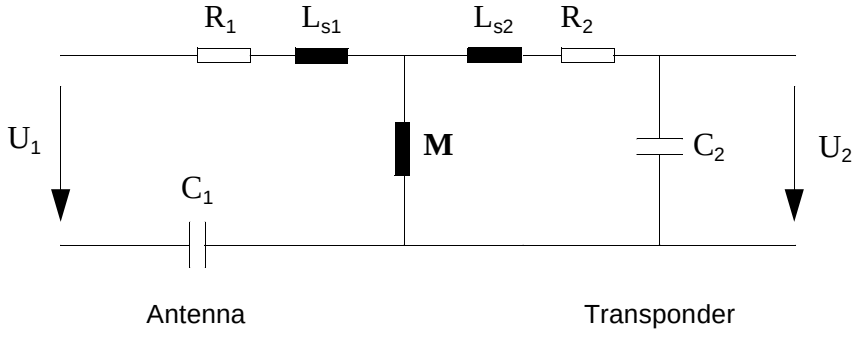


Fig.3 Introduction of transformer equivalent circuit

with

$$M = k\sqrt{L_1 L_2}$$

and

$$L_{s1,s2} = L_{1,2} - M$$

The following equivalent voltage source describes the electrical behaviour of the inductive link from the transponder circuit´s view. The describing equations to derive in the following include all relevant system parameters of basestation and inductive link; the behaviour of the transponder itself will be discussed in the following chapters based on figure 4 and the corresponding equations for U_0 and Z_i (4, 5; see below).

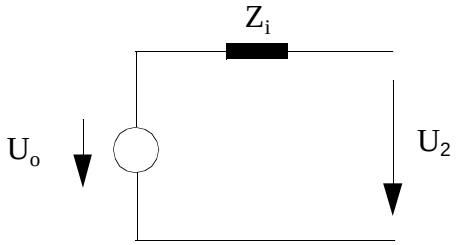


Fig.4 Equivalent circuit of inductive link

Complex values will not be underlined in the following to enhance readability. All variables may be considered complex unless otherwise stated, or obvious (e.g. resistances).

First, some general definitions are given; they are used frequently throughout this document:

$$\mathrm{par}(X,Y) = \frac{X \cdot Y}{X + Y}$$

$$XC = \frac{1}{j \cdot \omega \cdot C}$$

$$XL = j \cdot \omega \cdot L$$

$$Q = \frac{\omega L}{R}$$

Additionally, the following equations, which describe the detuning of a resonant circuit relative to the nominal frequency, are used in the following:

$$p_1 = 1 - \left(\frac{f_{r1}}{f_0}\right)^2 \quad \text{with} \quad f_{r1} = \frac{1}{2\pi\sqrt{L_1 C_1}}$$

$$p_2 = 1 - \left(\frac{f_{r2}}{f_0}\right)^2 \quad \text{with} \quad f_{r2} = \frac{1}{2\pi\sqrt{L_2 C_2}} \qquad (1)$$

L_1, C_1 denotes the basestation resonant circuit, L_2,C_2 that of the transponder, and f_0 is the nominal operating frequency as generated by the basestation driver stage (usually quartz stabilized).

The open circuit voltage U_0 of the circuit equates with U_M (denoting the voltage across M)

$$U_M = U_1 \cdot \frac{\mathrm{par}(XM, (R_2 + XL_{s2} + XC_2))}{R_1 + XL_{s1} + XC_1 + \mathrm{par}(XM, (R_2 + XL_{s2} + XC_2))}$$

and

$$U_0 = U_M \cdot \frac{XC_2}{R_2 + XL_{s2} + XC_2}$$

to

$$U_0 = U_1 \cdot \frac{XC_2}{R_2 + XL_{s2} + XC_2} \cdot \frac{par(XM, (R_2 + XL_{s2} + XC_2))}{R_1 + XL_{s1} + XC_1 + par(XM, (R_2 + XL_{s2} + XC_2))} \quad (2)$$

The output impedance of the equivalent voltage source follows when short-circuiting voltage source U_1:

$$Z_i = par(XC_2, (XL_{s2} + R_2 + par(XL_{s1} + R_1 + XC_1, XM))) \quad (3)$$

In order to discuss the system behaviour in dependence of component tolerances, equations (2) and (3) will be normalized to the carrier frequency that is generated by the basestation driver (usually 125 kHz). By introducing (1), and the quality factors Q_1,Q_2, the equivalent voltage source equates to

$$U_0 = U_1 \cdot k\sqrt{\frac{L_2}{L_1}} \cdot Q_1 \cdot Q_2 \cdot \frac{(1 - p_2)}{(1 + jQ_1 \cdot p_1) \cdot (1 + jQ_2 \cdot p_2) + Q_1 \cdot Q_2 \cdot k^2} \quad (4)$$

and

$$Z_i = \omega_0 L_2 \cdot (1 - p_2) \cdot \left(\frac{Q_2 \cdot (1 + jQ_1 \cdot p_1)\,(1 - p_2)}{(1 + jQ_1 \cdot p_1) \cdot (1 + jQ_2 \cdot p_2) + Q_1 \cdot Q_2 \cdot k^2} - j \right) \quad (5)$$

For an exemplary derivation of equations (4) and (5), see appendix 1.

2.2 Transponder integrated circuit

Now that an equivalent voltage source for the inductive link is set up, an equivalent circuit of the transponder integrated circuit module needs to be added to complete the electrical system description (figure 5).

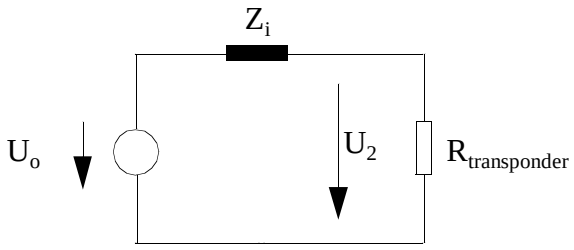


Fig.5 Equivalent circuit of inductive link, including transponder

Transient simulations have shown that the transponder circuitry can be adequately modelled as a complex impedance. The imaginary part is capacitive and essentially constant, and is taken into account during mass production by adjusting the value of C_2 to compensate a detuning of the resonant circuit. The remaining resistive part $R_{transponder}$ is subject of the following analysis, since it describes the energy consumption of the transponder circuit.

The transponder circuit operates internally by limiting the voltage across the antenna coil (U_2 in figure 5) to certain fixed levels which are dependent of the operating mode, the so-called “clipping levels”. Assuming in a first step that the inductive link (U_0, Zi, see equations (4), (5)) is able to supply the transponder circuit with enough energy so that the clipping level U_2 will be reached, the transponder equivalent resistance becomes a function of the system parameters:

$$R_{transponder} = f(U_0, Z_i, U_2)$$

We need to solve the voltage divider equation to derive a equation for $R_{transponder}$ (U_2 is in phase with $R_{transponder}$, so the absolute values can be used):

$$|U_2| = R_{transponder} \cdot \left| U_0 \cdot \frac{1}{Z_i + R_{transponder}} \right| \quad (6)$$

After squaring,

$$|U_2|^2 = |U_0|^2 \cdot \frac{R_{transponder}^2}{|Z_i + R_{transponder}|^2}$$

$$R_{transponder}^2 \cdot \left(|U_2|^2 - |U_0|^2 \right) + R_{transponder} \cdot 2 \cdot |U_2|^2 \cdot Real\{Z_i\} + |U_2|^2 \cdot |Z_i|^2 = 0$$

Only the positive square root solution of this quadratic equation is meaningful here:

$$R_{transponder} = \frac{Real\{Z_i\} \cdot |U_2|^2}{|U_0|^2 - |U_2|^2} + \sqrt{\left(\frac{Real\{Z_i\} \cdot |U_2|^2}{|U_2|^2 - |U_0|^2} \right)^2 - \frac{|U_2|^2 \cdot |Z_i|^2}{|U_2|^2 - |U_0|^2}} \quad (7)$$

The result may become negative or complex, which is the case when the inductive link is not able to reach the clipping voltage level. A fixed value which represents the actual power consumption of the transponder circuit will be substituted for $R_{transponder}$ in that case, since the voltage limiting circuit becomes inactive. A practical value of approximate 200kΩ has been measured.

2.3 Energy transmission to transponder

The data sheets of the Philips PCF79xx transponder family state a minimum ambient flux density for the transponder to operate, which is generally highest for Program-Mode operation. The relation between the flux density specification and the required electrical parameters of the system, to meet this demand, will be derived in the following.

At a fixed operating frequency, the induced voltage into the transponder coil U_{q2} is proportional to the applied flux density:

$$U_{q2} = \gamma \cdot B$$

The proportionality factor γ is mainly determined by the coil parameters (geometry, material). For the PCF79xx transponders packaged as “Stick” devices, an approximate typical value of

$$\gamma \approx 20 \frac{mV}{\mu T}$$

at a frequency of 125kHz has been measured.

Figure 6 shows the transponder antenna with the induced voltage U_{q2}, including the equivalent circuit of the transponder circuit as developed in section 2.2.

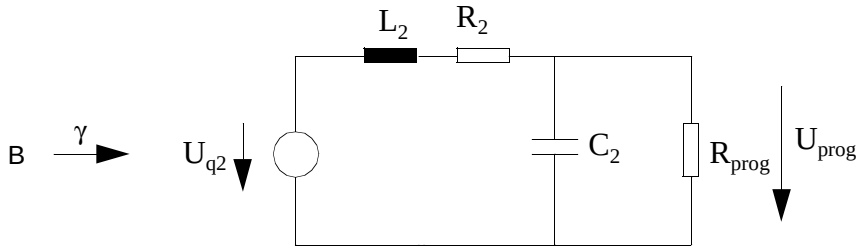


Fig.6 Energy transmission to transponder

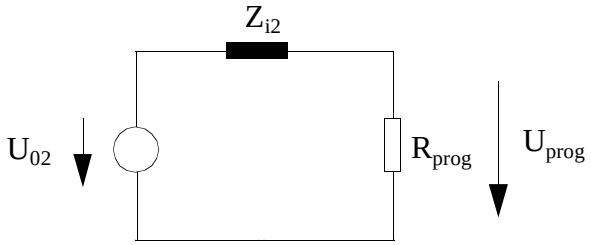


Fig.7 Equivalent circuit of figure 6

Note that the inductive link is *not* modelled using the transformer circuit in this case, since we do not regard the basestation side in the first step. Thus, the equivalent circuit of figure 6 as shown in figure 7 differs from that in figure 5; different symbols are used for the equivalent voltage source to avoid confusion here. Nevertheless, the equivalent circuit of the transponder module ($R_{transponder}$, U_2 in figure 5) is valid. The corresponding symbols in figure 6 are: R_{prog} which denotes the circuit equivalent resistance in programming mode, and U_{prog} which denotes the programming voltage that has to be reached. We declare

$$U_{prog} = |U_2|$$

since U_2 is a complex value, and U_{prog} is an absolute value that is given by the transponder IC. This leads to a modified transponder equivalent circuit equation (7):

$$R_{prog} = \frac{\mathrm{Real}\{Z_{i2}\} \cdot U_{prog}^{\,2}}{|U_{02}|^2 - U_{prog}^{\,2}} + \sqrt{\left(\frac{\mathrm{Real}\{Z_{i2}\} \cdot U_{prog}^{\,2}}{U_{prog}^{\,2} - |U_{02}|^2}\right)^2 - \frac{U_{prog}^{\,2} \cdot |Z_{i2}|^2}{U_{prog}^{\,2} - |U_{02}|^2}} \tag{8}$$

Together with the equations for the equivalent voltage source

$$U_{02} = U_{q2}\frac{1}{1 + j\omega R_2 C_2 - \omega^2 L_2 C_2}$$

$$Z_{i2} = \mathrm{par}\left(\frac{1}{j\omega C_2}, (j\omega L_2 + R_2)\right)$$

$$U_{02} = U_{q2} \cdot \frac{-j \cdot Q_2 \cdot (1 - p_2)}{1 + jQ_2 \cdot p_2} \tag{9}$$

$$Z_{i2} = \omega L_2 \cdot \frac{(Q_2 - j) \cdot (1 - p_2)}{1 + jQ_2 \cdot p_2} \tag{10}$$

we are able to set up the relation between the minimum ambient flux density specification from the transponder´s data sheet to the corresponding R_{prog}. It can be seen that

$$R_{prog} = f(B, U_{prog}, \text{Transponder antenna parameters})$$

The transponder´s minimum ambient flux density B from the data sheet is related to a transponder antenna that is detuned to its specified tolerance limits. R_{prog} will be computed for that condition (the fact that negative and positive antenna detunings lead to slightly different results will be neglected in the following). The resulting value will then denote the worst case energy consumption of the transponder $P_{prog} = (U_{prog})^2 \cdot R_{prog}$.

Figure 8 shows the relation between R_{prog} and B for three transponder types:

1. PCF7930XP (U_{prog} = 9V, Transponder antenna frequence tolerance = +- 3%)
2. PCF79735 (U_{prog} = 9V, Transponder antenna frequence tolerance = +- 2.5%)
3. PCF79736 (U_{prog} = 4V, Transponder antenna frequence tolerance = +- 3.2%)

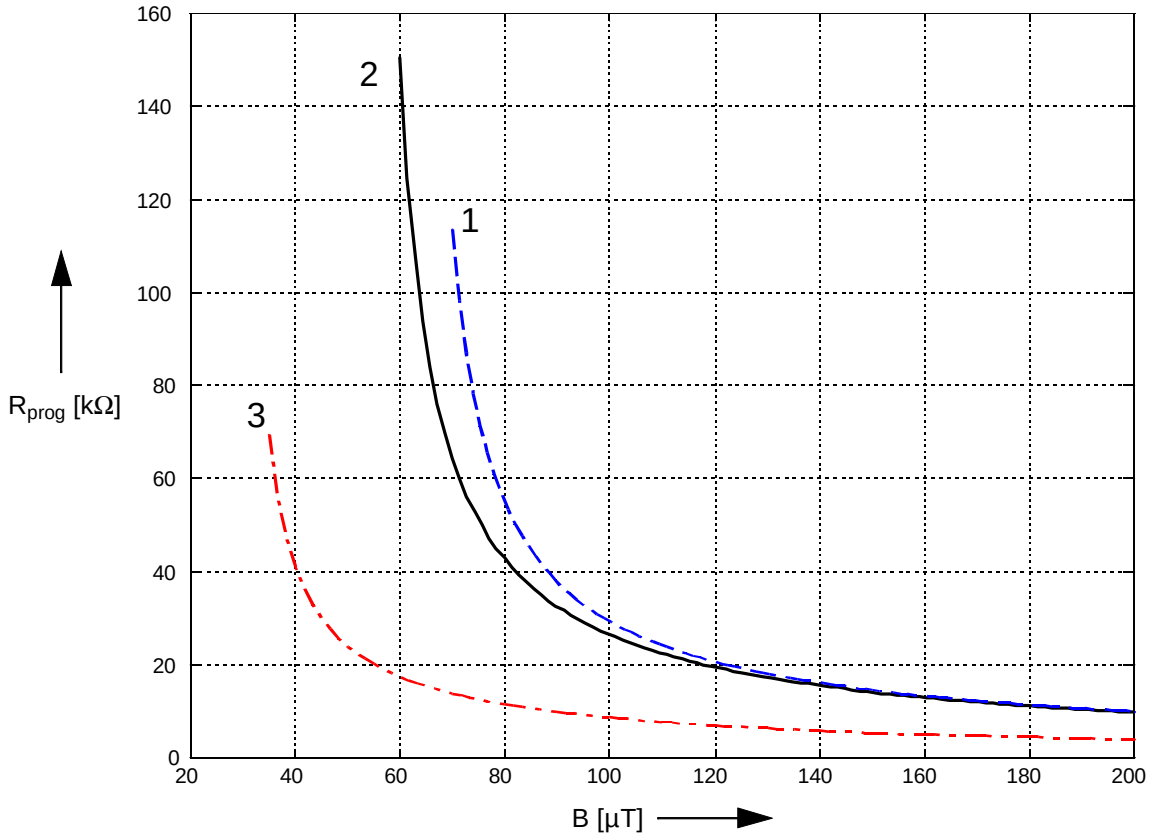


Fig.8 Equivalent transponder resistance versus programming flux density

For example, the specified value of B_{prog} = 100µT for the PCF79735, yields approx. R_{prog} = 25 kΩ.

As a last step, the computed value of R_{prog} will be inserted in the equivalent circuit of the inductive link (figure 5). The simulation program will then compute the value of U_2 using equations (4) and (5) while varying the detuning of basestation and transponder antennas. Each location in this two dimensional plane which fulfils the condition

$$|U_2| = U_{prog}$$

is a solution where the transponder operates precisely at its specified energy limit.

A example is shown in figure 9. The resonant frequency of the transponder antenna is varied in vertical direction, that of the basestation antenna in horizontal direction. Both axes show the deviation relative to the nominal frequency. The contour line encloses the area where the transponder´s energy limit is exceeded.

The two horizontal dotted lines denote the transponder´s specified resonant frequency tolerance limits. The minimum frequency tolerance requirement for the basestation antenna results from the intersection of the curve with these limits. The vertical dotted lines show the computed result of that; the hatched area that is set up by the four limits can be considered as a Safe Operating Area (SOAR).

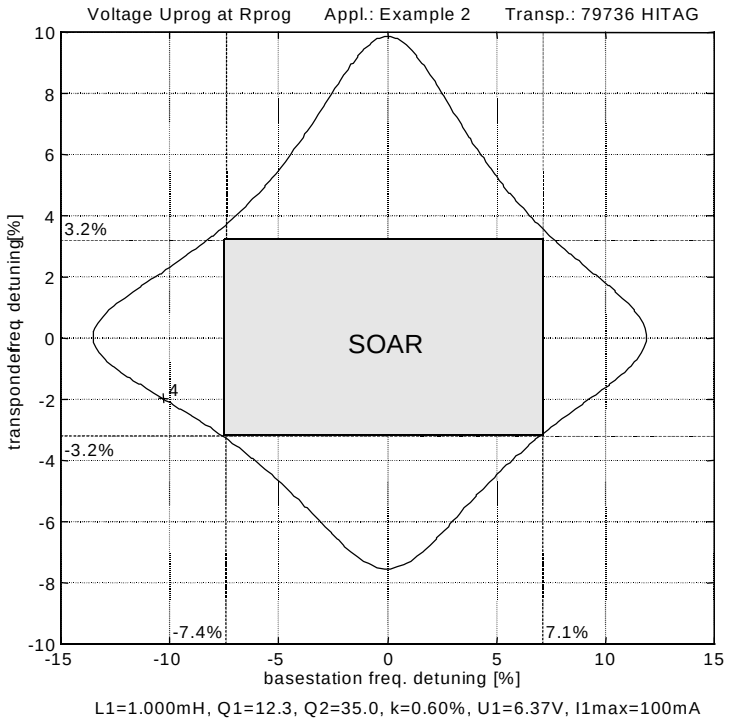


Fig.9 Example of energy limit simulation

2.4 Load modulation principle

While in Read-Mode, the transponder´s integrated circuit transmits binary data to the basestation by applying two different resistive loads to its antenna (see figure 10). The both load states can be expressed by a parallel circuitry of a basic load resistor $R_{bas1,2}$ and a voltage limiter $U_{clip1,2}$.

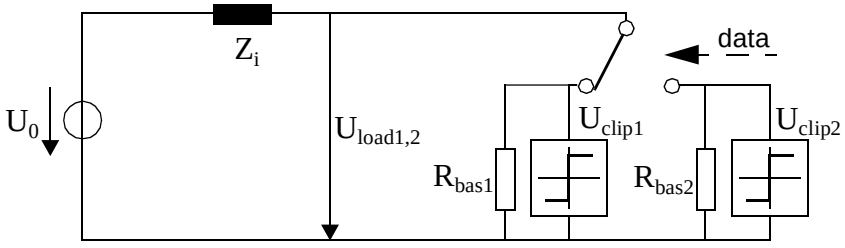


Fig.10 Transponder modulation model

In case that the amplitude is below the clipping voltage, the equivalent load resistor $R_{load1,2}$ is equal to the basic load resistor $R_{bas1,2}$. In the other case the load resistor is given by the equivalent resistor of the voltage limiter.

$$R_{load1,2} = \begin{cases} R_{bas1,2} & \text{if} \quad U_{load1,2} < U_{clip1,2} \\ R_{clip1,2} & \text{else} \end{cases} \tag{11}$$

This ascertains two separate equivalent resistances $R_{load1,2}$ for the transponder circuit, one for each binary state. The resistance of the voltage limiter $R_{clip1,2}$ can be calculated employing equation (7) with $U_{clip1,2} = |U_2|$:

$$R_{clip1,2} = \frac{Real\{Z_i\} \cdot U_{clip1,2}^2}{|U_0|^2 - U_{clip1,2}^2} + \sqrt{\left(\frac{Real\{Z_i\} \cdot U_{clip1,2}^2}{U_{clip1,2}^2 - |U_0|^2}\right)^2 - \frac{U_{clip1,2}^2 \cdot |Z_i|^2}{U_{clip1,2}^2 - |U_0|^2}} \quad (12)$$

Regarding the equivalent voltage source as described in section 2.1, refer to the definitions of U_0 and Z_i in equations (2) and (3).

In order to compute the the transponder modulation to the basestation side, the inductive link will be considered again now (see figure 11). The load modulation which the transponder performs by switching between R_{load1} and R_{load2} is recognized as a change in the complex impedance of the inductive link at the basestation side.

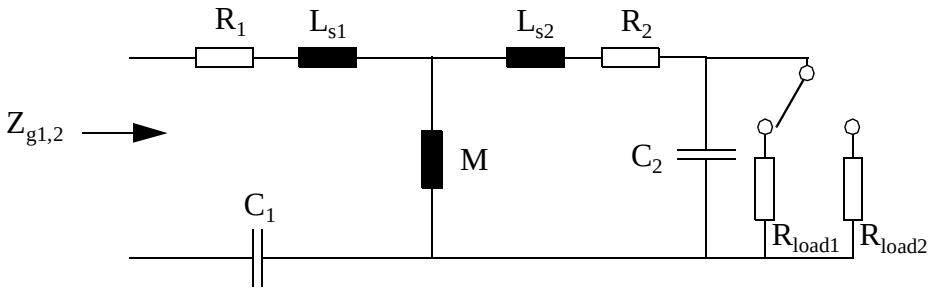


Fig.11 Simplified equivalent circuit for read-mode

The impedances Z_{g1} and Z_{g2} that correspond to the two resistances R_{load1} and R_{load2} result in:

$$Z_{g1,2} = R_1 + j\omega L_{s1} + \frac{1}{j\omega C_1} + par\left(j\omega M, j\omega L_{s2} + R_2 + par\left(\frac{1}{j\omega C_2}, R_{load1,2}\right)\right)$$

Introducing relative coefficients equation (1), the impedances can be described as:

$$Z_{g1,2} = R_1 \cdot \left(1 + jQ_1 \cdot p_1 + \left(\frac{Q_1 \cdot Q_2 \cdot k^2}{1 + jQ_2 \cdot p_2 + Q_2 \cdot \frac{R_{load1,2} \cdot (1 - p_2)}{\omega_0 L_2 \cdot (1 - p_2) + j \cdot R_{load1,2}}}\right)\right) \quad (13)$$

The Philips PCF79x30, PCF79x31 and PCF79x35 transponders perform this modulation by applying two different clipping levels $U_{clip1,2}$ to the transponder antenna as described in section 2.2. The voltage U_{clip1} is assigned to the upper clipping level, which protects the IC input against overvoltage. The switching to the lower clipping level (U_{clip2}) performs the load modulation. Both basic load resistances R_{bas1}, R_{bas2} represent only the power consumption of the IC without the clipping circuitry.

The PCF79x36 Transponder uses the same kind of overvoltage protection. The coil voltage is limited by an upper clipping level U_{clip1} in parallel to the basic load resistance R_{bas1}. But the load modulation is performed in a different way. In opposite to the pure clipping modulation, the circuitry of load state 2 is an actual combination of a limiter and a load resistor. This circuitry employs different components for the positive and negative half-wave:

Uclip2a, Rbas2a: Clipping voltage and basic load resistance during the positive half-wave

Uclip2b, Rbas2b: Clipping voltage and basic load resistance during the negative half-wave

The both load states can merged to anequivalent resistor which is calculated using the average loading voltage.

The equivalent loading resistances has to be calculated in following steps:

1. Calculation of the transponder voltage U_{2a} for the positive half-wave with R_{bas2a}:

$$U_{2a} = \left| U_0 \cdot \frac{R_{load2a}}{Z_i + R_{load2a}} \right|$$

   U_0 and Z_i are given in (2) and (3).

2. Limitation of U_{2a} to the clipping voltage U_{clip2a}.
3. Calculation of the transponder voltage U_{2b} for the negative half-wave with R_{bas2b}:

$$U_{2b} = \left| U_0 \cdot \frac{R_{load2b}}{Z_i + R_{load2b}} \right|$$

4. Limitation of U_{2b} to the clipping voltage U_{clip2b}.
5. Calculation of the mean transponder amplitude:

$$U_{2ab} = \frac{U_{2a} + U_{2b}}{2}$$

6. Now, the equivalent resistor to this voltage is given using equation(12) with inserting U_{2ab} instead of $U_{clip1,2}$.

$$R_{load2} = \frac{Real\{Z_i\} \cdot U_{2ab}^2}{|U_0|^2 - U_{2ab}^2} + \sqrt{\left(\frac{Real\{Z_i\} \cdot U_{2ab}^2}{U_{2ab}^2 - |U_0|^2} \right)^2 - \frac{U_{2ab}^2 \cdot |Z_i|^2}{U_{2ab}^2 - |U_0|^2}}$$

2.5 Demodulation principles

For important demodulation principles will be discussed throughout the following sections:

- Envelope demodulation,
- Phase demodulation, and
- Quadrature demodulation.
- Adaptive Sampling Time demodulation (AST)

As a first step, we will have a closer look at equation (12), which describes the behaviour of the transponder modulation with respect to detuned basestation and transponder antennas. Figure 12 shows a set of curves in the complex plane of $Z_{g1,2}$ for one specific basestation antenna detuning. The overview plot in the lower left shows the impedance of the basestation antenna of approx. 80 + j70 Ω. (A basestation antenna resonant frequency detuning of -10% was chosen for this plot.) The modulation lies “on top” of this vector and is shown enlarged in the upper right. Z_{g1} (´+´ symbol) and Z_{g2} (´o´ symbol) have been computed for various detuning points of the transponder antenna based on equation (12). Each pair of impedance “states” has been connected by a line. Each line can be regarded as the visualization of Z_{g2} - Z_{g1}, which denotes the relative modulation vector for a specific transponder antenna detuning.

It can be seen that there are certain antenna detuning points where the modulation vector has no imaginary part, whereas there are others with a zero real part. Additionally, the modulation direction reverses also with greater antenna detuning. Still, the amplitude remains existent but decreases with increasing transponder antenna detuning

.

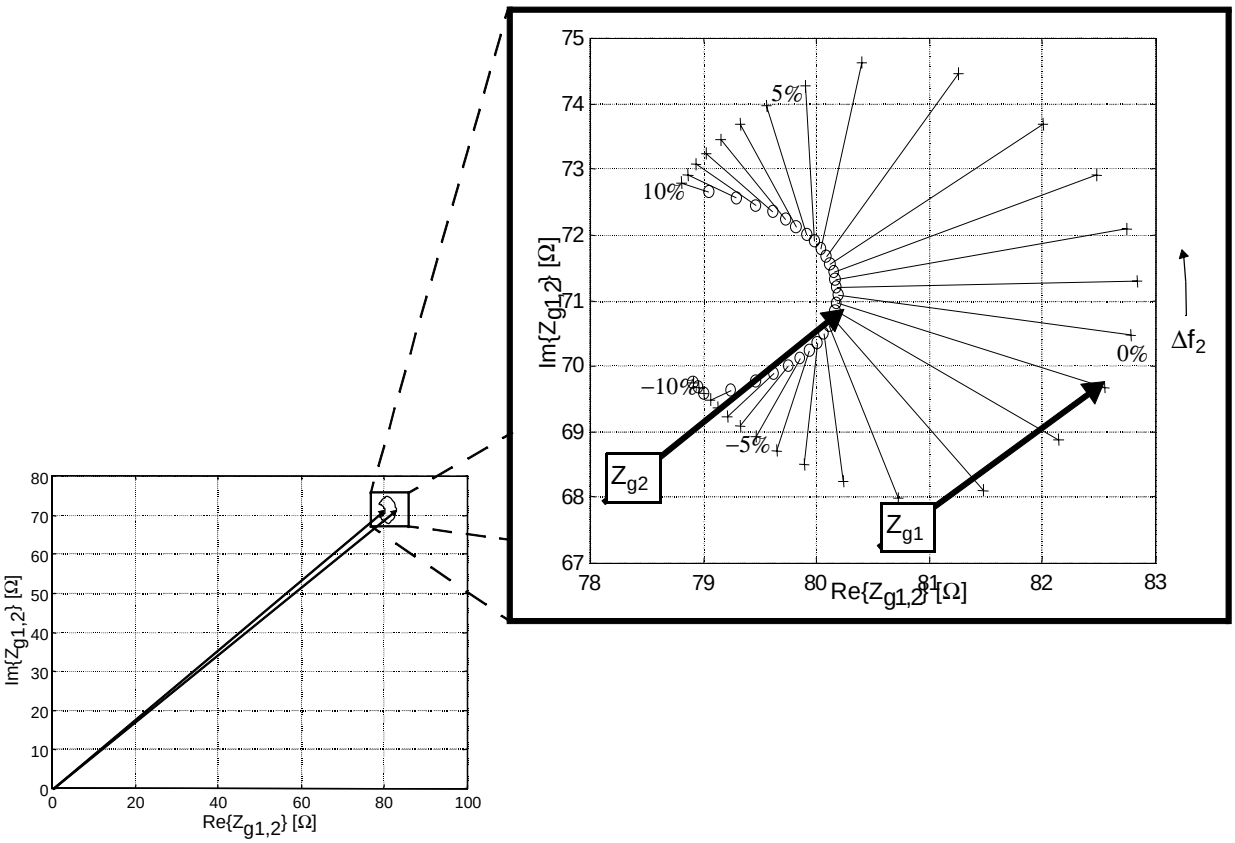


Fig.12 System impedances in Read-Mode

2.5.1 Envelope demodulator

The most common approach to demodulate the described change in $Z_{g1,2}$ is the envelope demodulator. A design example is shown in figure 13. The demodulator input is the voltage across the basestation resonant capacitor C_1 (see figures 1 and 2), which equates to

$$U_{C1\,(1,2)} = U_1 \cdot \frac{1}{j\omega C_1} \cdot \frac{1}{Z_{g1,2}}$$

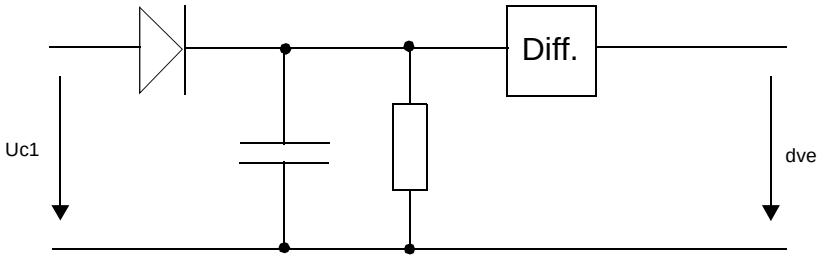


Fig.13 Envelope demodulator

The demodulator consists basically of a one-way rectifier followed by a differentiator “Diff”. Typically, the output signal will be amplified, filtered and digitized then.

The rectifier circuit outputs the amplitude, or absolute value, of its input signal; the differentiator then generates the demodulated signal Δv_e, which equates to

$$\Delta v_e = |U_{C1(2)}| - |U_{C1(1)}| = U_1 \cdot \left|\frac{1}{j\omega C_1}\right| \left(\left|\frac{1}{Z_{g2}}\right| - \left|\frac{1}{Z_{g1}}\right| \right) \quad (14)$$

The demodulation principle has certain restrictions with respect to detunings of basestation and transponder antennas, which will be described in the following.

Figure 14 shows the effect of detuning the basestation antenna to the direction of the $Z_{g1,2}$ vector (see also figure 12). Depending on the detuning direction, the vector will have a positive or negative imaginary component.

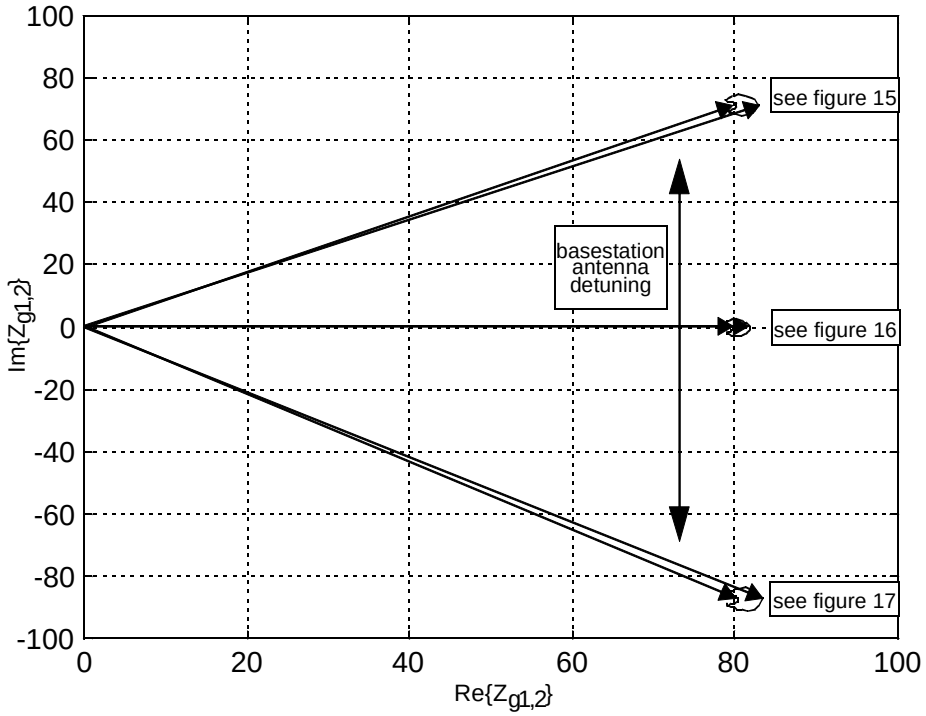


Fig.14 Influence of basestation antenna detuning to $Z_{g1,2}$

Figure 15 shows an enlarged section of figure 14. The thick lines show certain transponder antenna detuning points at which the absolute value of Z_{g1} and Z_{g2} are equal. The demodulator will fail under these operating conditions, although the modulation amplitude itself

$$\Delta Z = |Z_{g2}| - |Z_{g1}|$$

would still be sufficient. Also noticeable is the change in modulation direction between the light and dark shaded detuning areas, which will cause an inverted demodulator output signal at strong transponder antenna detunings.

Comparing figures 15 to 17, one recognizes that specific points of transponder antenna detuning where described zero demodulation occurs also depend on the basestation antenna detuning.

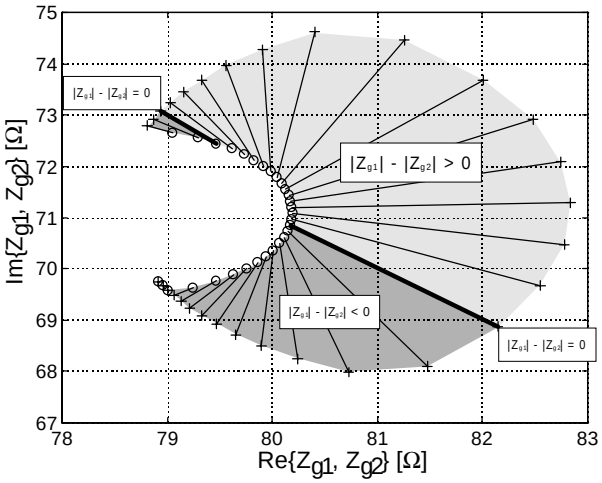


Fig.15 $Z_{g1,2}$ at -4.6% basestation antenna detuning

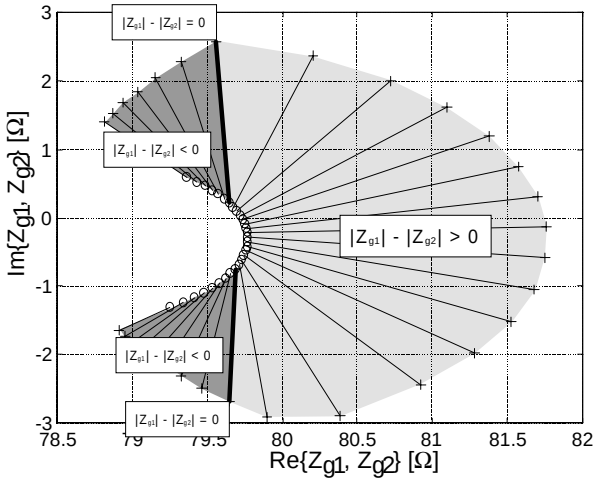


Fig.16 $Z_{g1,2}$ at 0% basestation antenna detuning

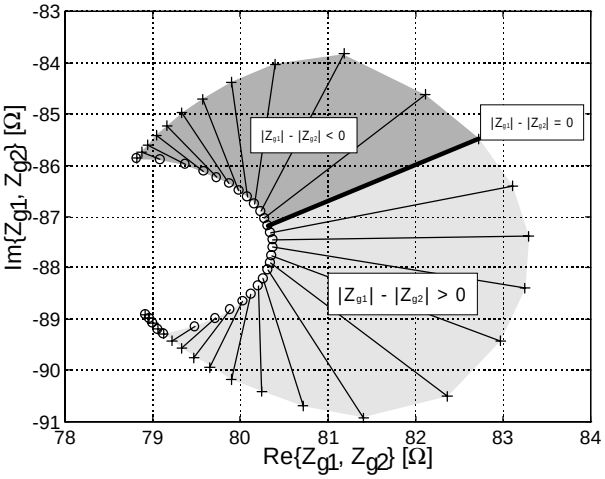


Fig.17 $Z_{g1,2}$ at 4.6% basestation antenna detuning

Figure 18 shows a three dimensional plot which visualizes the above described effect. The independent axes X and Y describe the detuning of the basestation and transponder resonant antenna circuits. For each grid point of this two dimensional surface, the result of equation (14) is evaluated and plotted in the Z axis, yielding a three dimensional mesh grid.

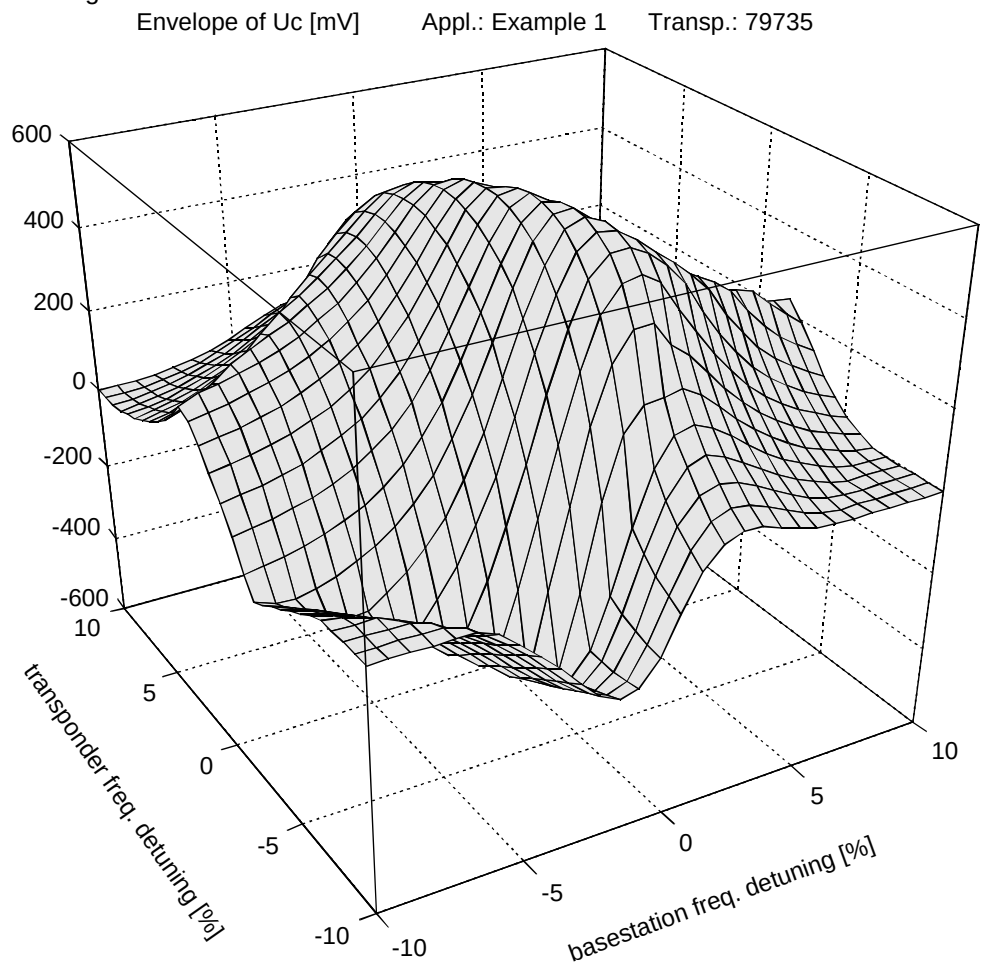
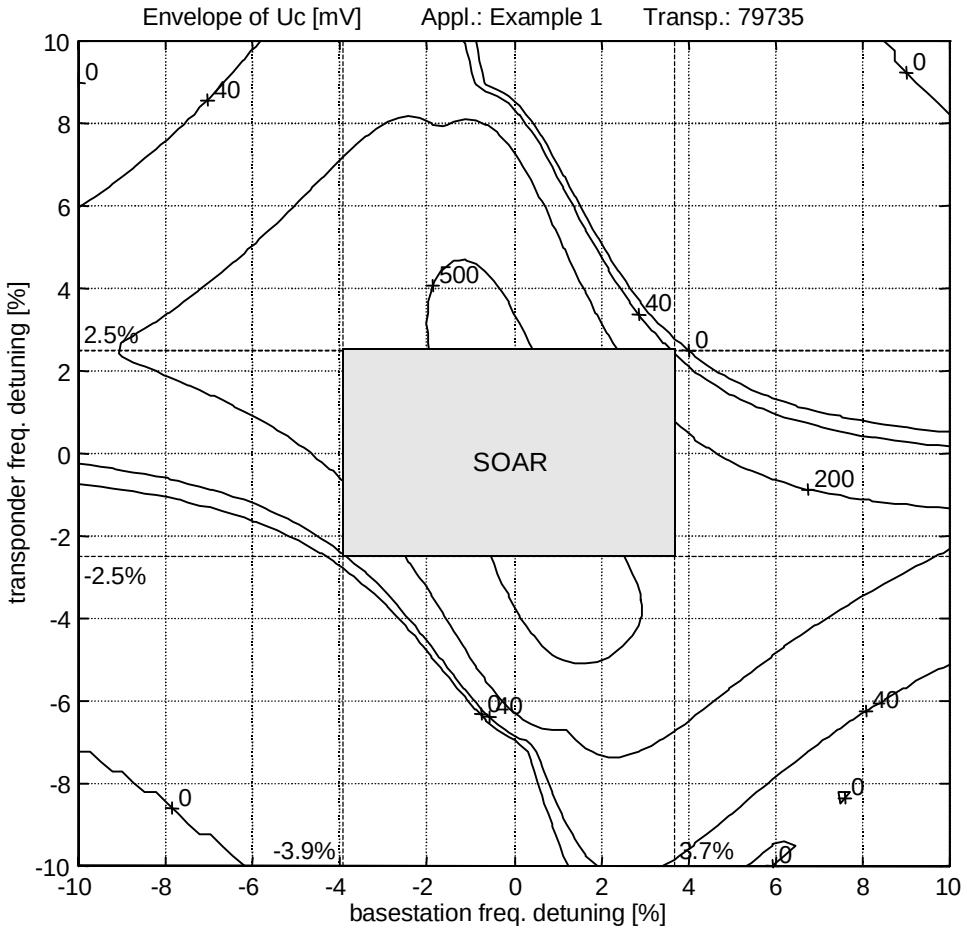


Fig.18 3D plot of tolerance field simulation with envelope demodulator

When advancing from the center of the plot, with the maximum sensed voltage, to the point of detuning where both basestation and antenna are operating at a resonant frequency of f_0 -10%, it can clearly be seen how the demodulated voltage decreases and crosses zero, followed by a negative valley until it finally becomes zero at (-10%, -10%) detuning.

A more convenient method of displaying the demodulation voltage is the contour plot as shown in figure 19. The simulation program draws a number of curves with equal demodulated voltage; the course of zero modulation can be traced easily here. A Safe Operation Area (SOAR) has been inserted into the plot, assuming a demodulator sensitivity of 40 mV.



L1=1.000mH, Q1=15.0, Q2=35.0, k=1.00%, U1=6.37V, I1max=121mA

Fig.19 Contour plot of tolerance field simulation with envelope demodulator

2.5.2 Phase demodulator

The phase demodulator senses the phase of U_{C1} relative to the driver´s output voltage U_1:

$$v_{p(1,2)} = \arg(U_{c1(1,2)}) - \arg(U_1) \quad \text{with}$$

$$\arg(x) = \arctan\left(\frac{\mathrm{Imag}\{x\}}{\mathrm{Real}\{x\}}\right)$$

Using the same differentiating setup as described in section 2.5.1, the demodulated angle equates to:

$$\Delta v_p = \arctan\left(\frac{\mathrm{Imag}\{U_1 \cdot \frac{1}{j\omega C_1} \cdot \frac{1}{Z_{g2}}\}}{\mathrm{Real}\{U_1 \cdot \frac{1}{j\omega C_1} \cdot \frac{1}{Z_{g2}}\}}\right) - \arctan\left(\frac{\mathrm{Imag}\{U_1 \cdot \frac{1}{j\omega C_1} \cdot \frac{1}{Z_{g1}}\}}{\mathrm{Real}\{U_1 \cdot \frac{1}{j\omega C_1} \cdot \frac{1}{Z_{g1}}\}}\right) \quad (15)$$

Figure 20 shows a sample plot of the simulated demodulation. Noticeable is the zero modulation line which crosses the center of the plot where both antennas are not detuned, because the transponder´s load modulation contains no imaginary part in that case (compare figures 15 to 17).

However, it can also be noticed that modulation is detectable in those areas where the envelope demodulator has its zero modulation lines. When both demodulators would be used in conjunction, each would compensate the lack of the other, removing the zero lines and expanding the Safe Operating Area.

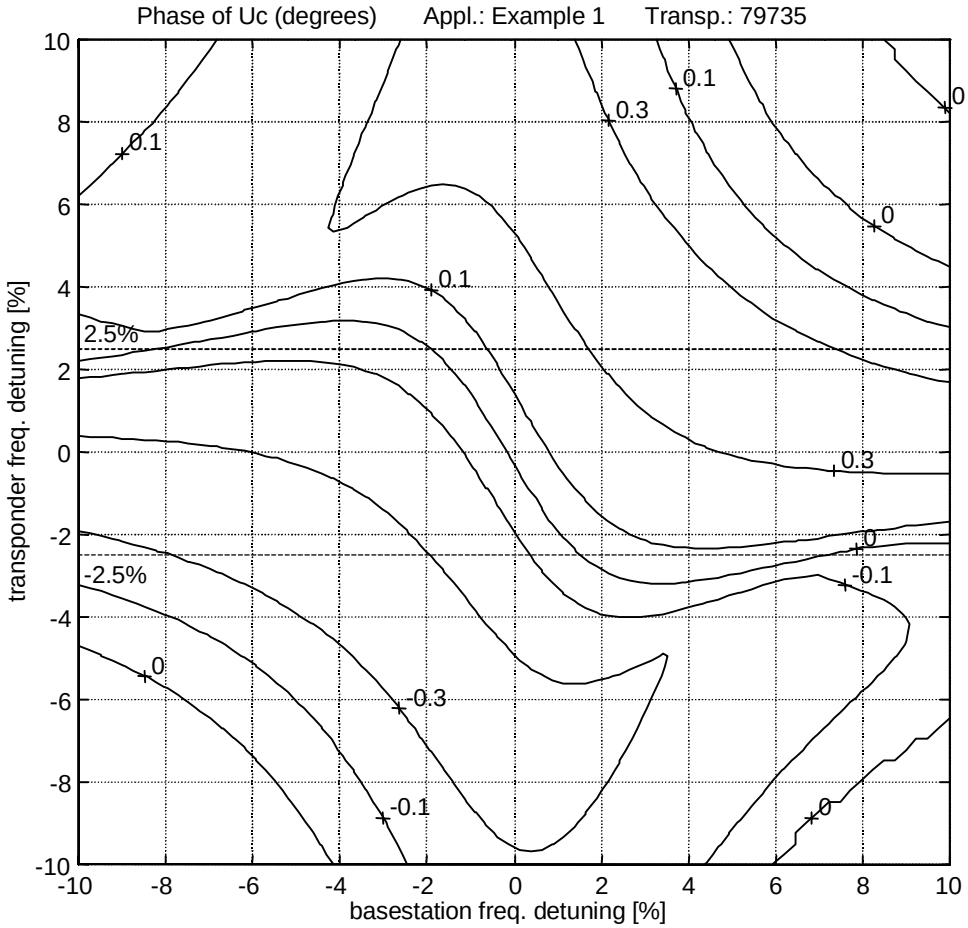


Fig.20 Tolerance field with phase demodulation

2.5.3 Quadrature demodulator

The quadrature demodulator consisting of two mixer circuits followed by low pass filters (figure 21) derives both the real and imaginary part of the modulated voltage U_{C1} across C_1:

$$U_{c1\,(1,\,2)} = \mathrm{Real}\{U_{c1\,(1,\,2)}\} + j \cdot \mathrm{Imag}\{U_{c1\,(1,\,2)}\}$$

The standard quadrature demodulator circuit has been modified by adding two differentiators, which will yield the real and imaginary parts of the modulation across C_1:

$$\Delta U_{c1} = \Delta v_r + j \cdot \Delta v_i \quad \text{with}$$

$$\Delta v_r = \mathrm{Real}\{U_1 \cdot \frac{1}{j\omega C_1} \cdot \frac{1}{Z_{g2}} - U_1 \cdot \frac{1}{j\omega C_1} \cdot \frac{1}{Z_{g1}}\} \quad \text{and}$$

$$\Delta v_i = \mathrm{Imag}\{U_1 \cdot \frac{1}{j\omega C_1} \cdot \frac{1}{Z_{g2}} - U_1 \cdot \frac{1}{j\omega C_1} \cdot \frac{1}{Z_{g1}}\} \qquad (16)$$

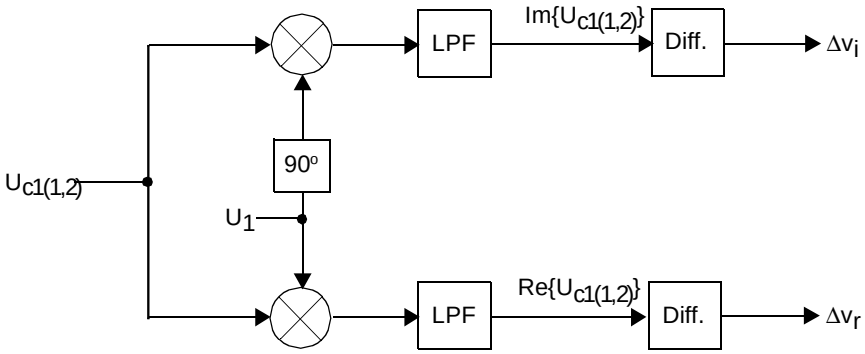


Fig.21 Quadrature demodulator

Figures 22 and 23 show example simulations of Δv_r and Δv_i. Both signals have zero modulation lines at that antenna detuning points which cause the modulation to become completely real (zero Δv_i) or completely imaginary (zero Δv_r). The quadrature demodulator is in fact a more systematic approach compared to the amplitude and phase demodulator pair. (Compare figure 22 with 20 and figure 23 with 19.) The problem of having two signals that each represent the same information, but not necessarily both being present the same time, can be solved by two different approaches. The more general (but in practice difficulty to implement) step is to derive the absolute value of Δv:

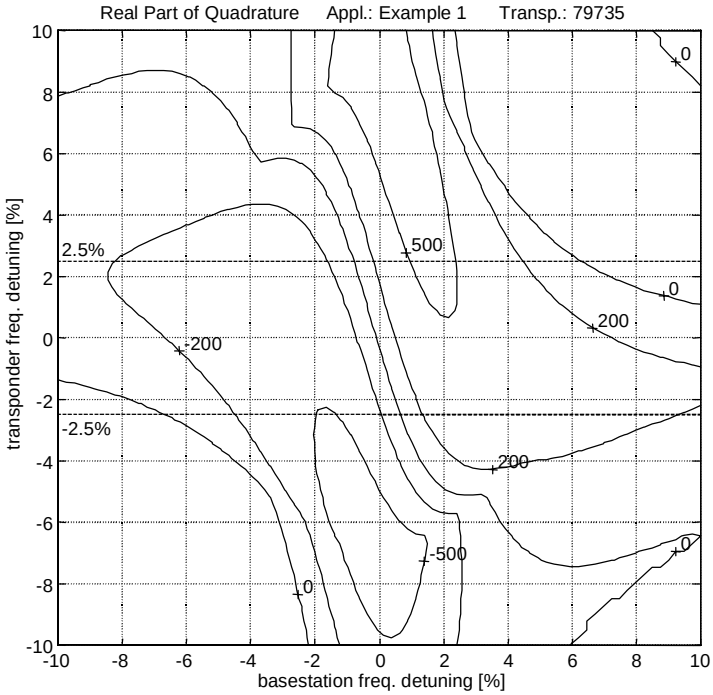
$$\Delta v_{abs} = \sqrt{\Delta v_r^2 + \Delta v_i^2}$$

The resulting simulation is shown in figure 24; no zero modulation lines can be observed, the Safe Operating Area exceeds the simulated range of +/-10% basestation antenna resonant frequency detuning.

A more practical solution is to choose the better signal by comparing the amplitudes of Δv_r against Δv_i and choosing the larger signal to output:

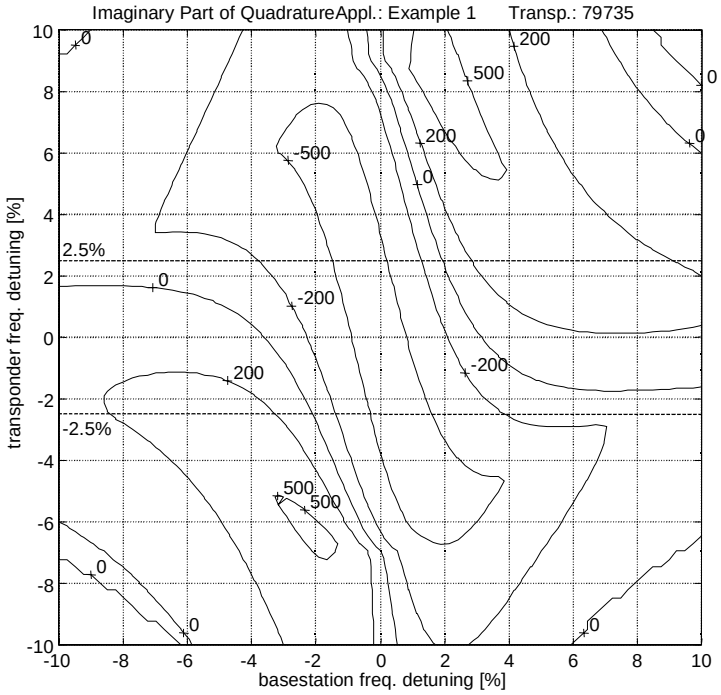
$$\Delta v_{max} = \max(\Delta v_r, \Delta v_i)$$

A plot of Δv_{max} is shown in figure 25.



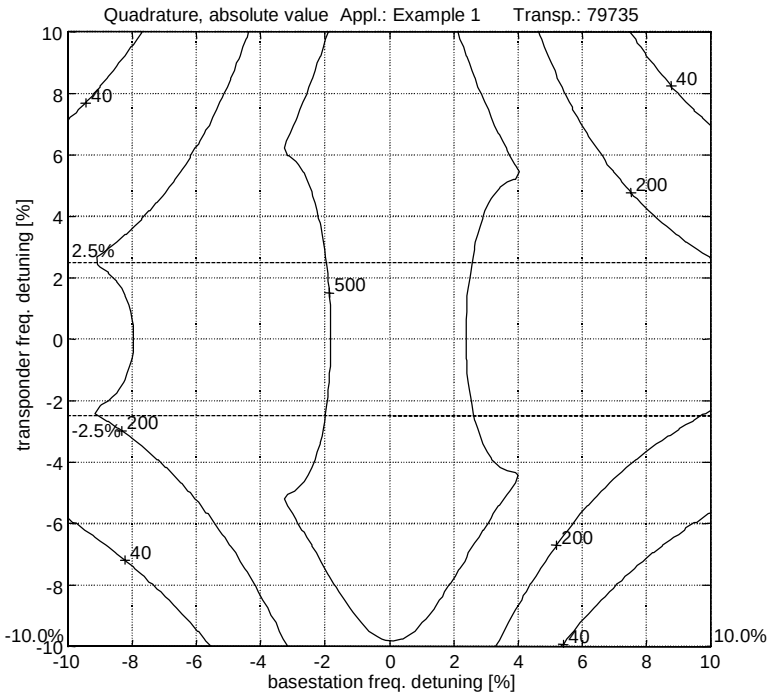
L1=1.000mH, Q1=15.0, Q2=35.0, k=1.00%, U1=6.37V, I1max=121mA

Fig.22 Quadrature demodulator output, real part



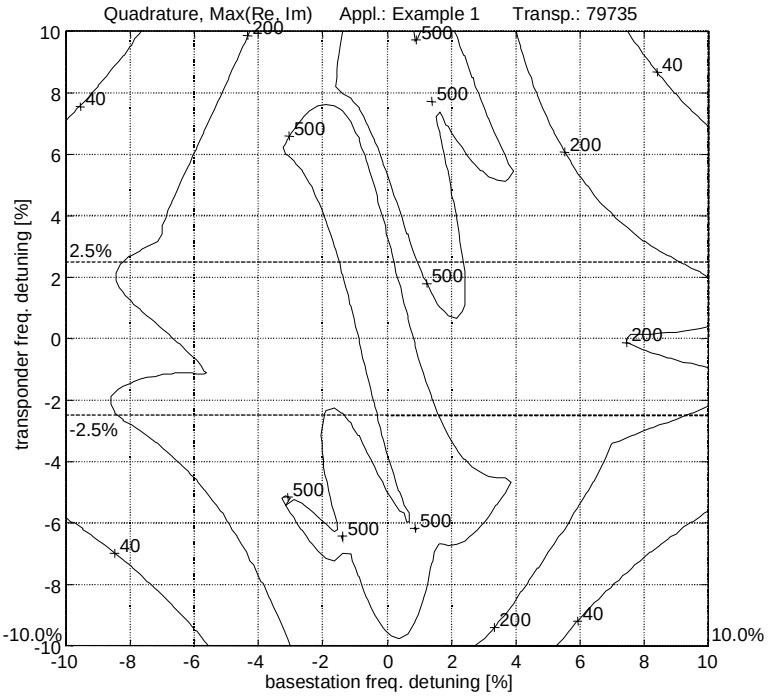
L1=1.000mH, Q1=15.0, Q2=35.0, k=1.00%, U1=6.37V, I1max=121mA

Fig.23 Quadrature demodulator output, imaginary part



L1=1.000mH, Q1=15.0, Q2=35.0, k=1.00%, U1=6.37V, I1max=121mA

Fig.24 Quadrature demodulator output, absolute value



L1=1.000mH, Q1=15.0, Q2=35.0, k=1.00%, U1=6.37V, I1max=121mA

Fig.25 Quadrature demodulator output, maximum decision

2.5.4 Adaptive sampling time demodulation

Background

The figures 15 to 17 show that the detuning of antenna resonant circuit (L_1, C_1) results in an asymmetrical transponder tolerance range for envelope demodulation. If only the real part of impedance $Z_{g1,2}$ is considered, the transponder tolerance yields a symmetrical range independent of the antenna detuning.

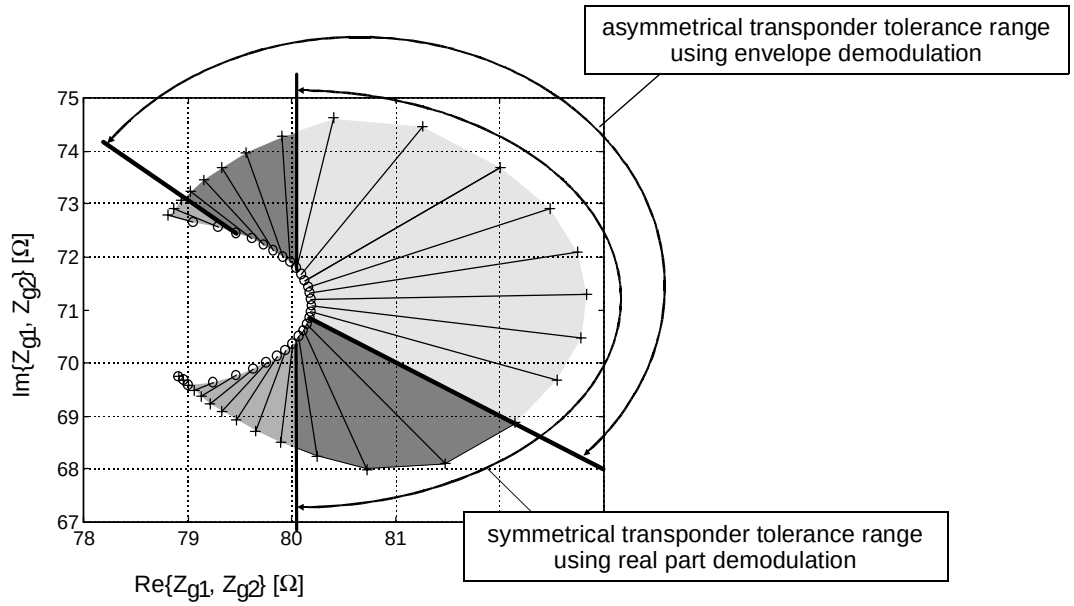


Fig.26 Real part demodulation of $Z_{g1,2}$ at -4.6% basestation antenna detuning

A proper definition is given by

$$\Delta v_{zr} = \frac{Real\{Z_{g1}\}}{|Z_g|} - \frac{Real\{Z_{g2}\}}{|Z_g|} = \frac{Real\{Z_{g1}\} + Real\{Z_{g2}\}}{|Z_g|} = \frac{Real\{Z_{g1} - Z_{g2}\}}{|Z_g|} = \frac{Real\{\Delta Z_g\}}{|Z_g|} \quad (17)$$

where Z_g denotes the antenna impedance without the transponder influences simulation result in figure 27 confirms the symmetrical tolerance field. Compared to the envelope demodulation in figure 20, the Safe Operating Area becomes an enlarged range.

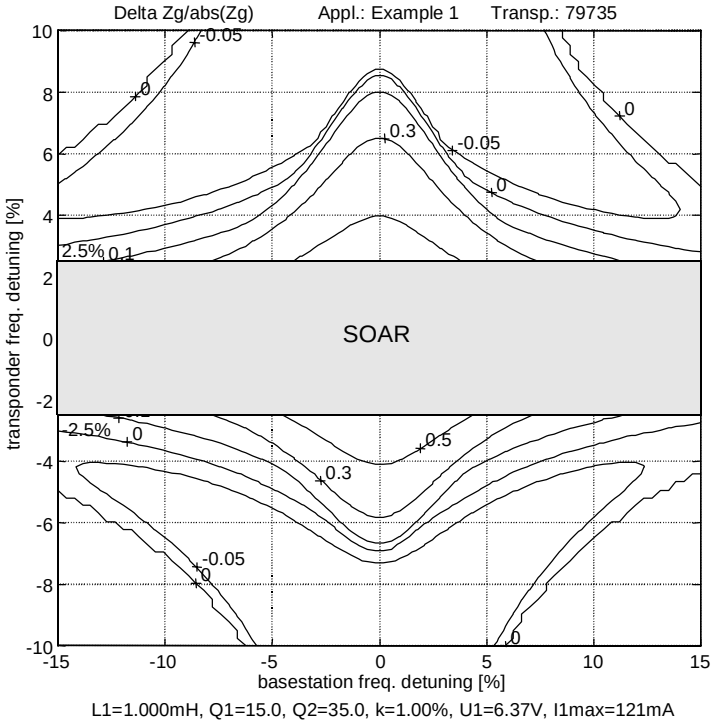


Fig.27 Real part difference of the antenna input impedance

In general, the impedance Z_g cannot be sensed directly. The voltage U_{C1} at the antenna capacitor is usually used as demodulator input. The relation between U_{C1} and $Z_{g1,2}$

$$U_{C1} = U_1 \cdot \frac{XC_1}{Z_{g1,2}}$$

shows, that the voltage deviation is the inverse ratio to the impedance $Z_{g1,2}$. The voltage deviation at C_1 is defined as ΔU_{C1}:

$$\begin{aligned}\Delta U_{C1} &= U_1 \cdot XC_1 \cdot \left(\frac{1}{Z_{g1}} - \frac{1}{Z_{g2}}\right) \\ &= U_1 \cdot XC_1 \cdot \left(\frac{Z_{g2} - Z_{g1}}{Z_{g1} \cdot Z_{g2}}\right) \\ &= U_1 \cdot XC_1 \cdot \frac{-\Delta Z_g}{Z_{g1} \cdot Z_{g2}}\end{aligned}$$

With

$$Z_{g1} = |Z_{g1}| \cdot e^{j\varphi_1} \quad \text{and} \quad Z_{g2} = |Z_{g2}| \cdot e^{j\varphi_2}$$

the deviation of U_{C1} is:

$$\Delta U_{C1} = U_1 \cdot XC_1 \cdot \frac{-\Delta Z_g}{|Z_{g1}| \cdot |Z_{g2}|} \cdot e^{-j(\varphi_1 + \varphi_2)} \tag{18}$$

With the approximation

$$|Z_{g1}| \approx |Z_{g2}| \approx |Z_g| \quad \text{and} \quad \varphi_1 \approx \varphi_2 \approx \varphi \tag{19}$$

the equation (18) can be rearranged to:

$$\frac{\Delta Z_g}{|Z_g|} = -\frac{|Z_g|}{U_1 \cdot X_{C1}} \cdot \Delta U_{C1} \cdot e^{j(2\varphi)}$$

The real part selection gives Δv_{zr} (see equation (17)):

$$\Delta v_{zr} = \frac{Real\{\Delta Z_g\}}{|Z_g|} = -Real\left\{\frac{|Z_g|}{U_1 \cdot X_{C1}} \cdot \Delta U_{C1} \cdot e^{j(2\varphi)}\right\}$$

Inserting $XC_1 = 1/(j\omega C1)$ yields:

$$\Delta v_{zr} = -Real\left\{\frac{|Z_g| \cdot j \cdot \omega \cdot C_1}{U_1} \cdot \Delta U_{C1} \cdot e^{j(2\varphi)}\right\}$$

$$= -\frac{|Z_g| \cdot \omega \cdot C_1}{U_1} \cdot Real\{j \cdot \Delta U_{C1} \cdot e^{j(2\varphi)}\}$$

With $j = -e^{-j \cdot \frac{\pi}{2}}$, this relation is:

$$\Delta v_{zr} = \frac{|Z_g| \cdot \omega \cdot C_1}{U_1} \cdot Real\left\{\Delta U_{C1} \cdot e^{j\left(2\varphi - \frac{\pi}{2}\right)}\right\}.$$

A rearrangement of this equation to

$$Real\left\{\Delta U_{C1} \cdot e^{j\left(2\varphi - \frac{\pi}{2}\right)}\right\} = \frac{U_1}{|Z_g| \cdot \omega \cdot C_1} \cdot \Delta v_{zr}$$

gives a proportional term to Δv_{zr} (see equation (17) and the corresponding simulation result in figure 27). The scaling coefficient

$$\frac{U_1}{|Z_g| \cdot \omega \cdot C_1}$$

is always real and positive. Thus, the simulation result of

$$\Delta v_{Real} = Real\left\{\Delta U_{C1} \cdot e^{j\left(2\varphi - \frac{\pi}{2}\right)}\right\} \quad (20)$$

in figure 28 displays the same location of zero contour lines (Δv_{Real}=0) as the zero contour lines of Δv_{zr}=0 shown in figure 27.

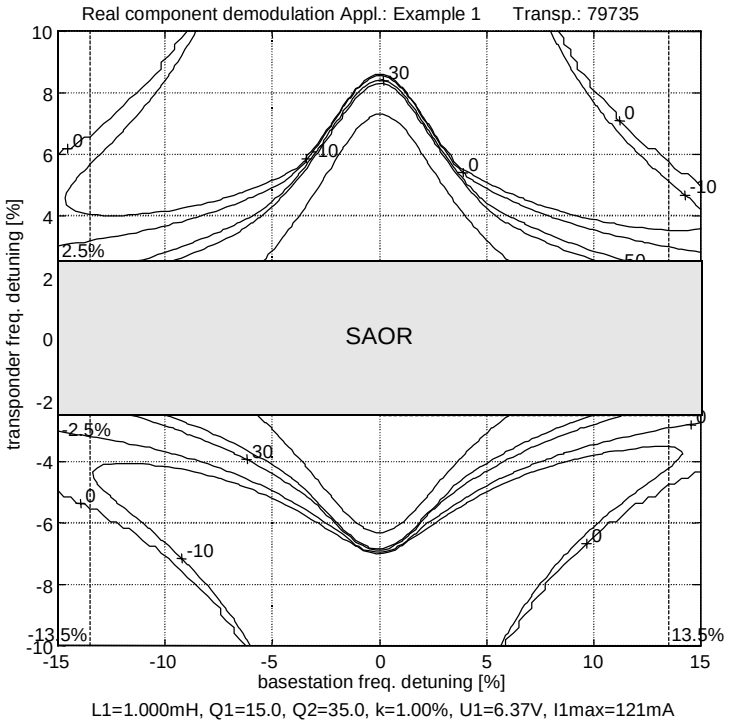


Fig.28 Real part demodulation

Sampling demodulation

The result of equation (20) makes known, that the phase of capacitor voltage needs to be shifted by

$$\rho_s = 2\varphi - \frac{\pi}{2} \tag{21}$$

before the real part is selected. The real part can be recovered by synchronous or coherent demodulation. The block diagram is shown in figure 29. A phase detector delivers the phase φ of the antenna input impedance which is used to adjust the phase shifter. The real part is selected by synchronous detector consisting of a multiplier and a low pass filter (see [1]).

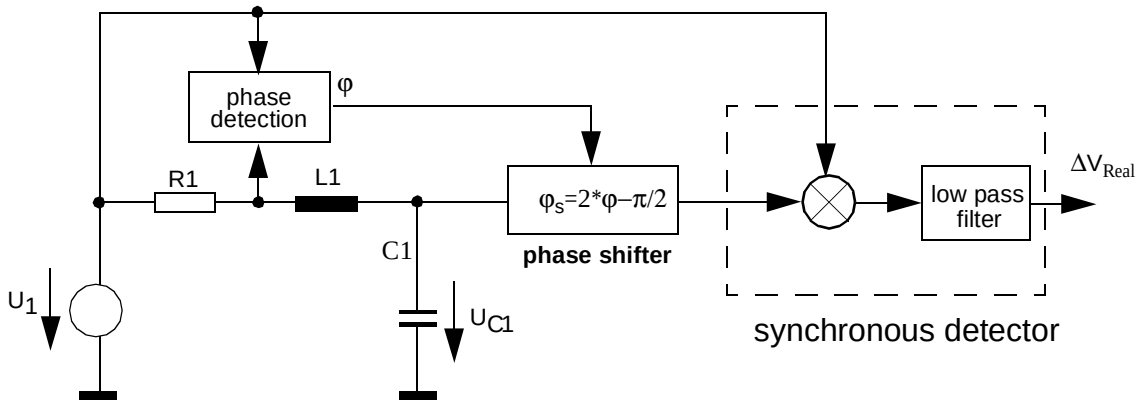
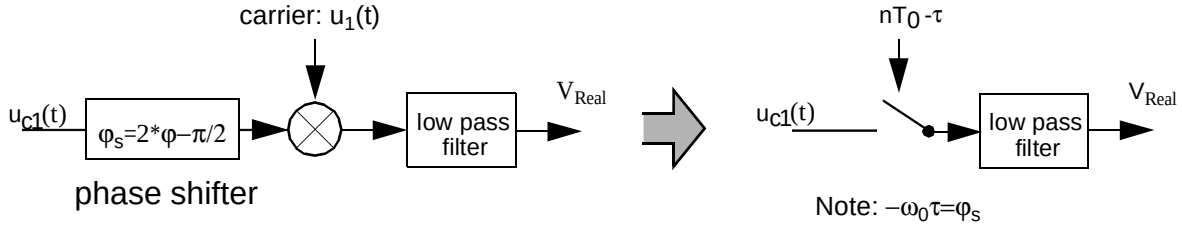


Fig.29 Real component detector

An alternative realization of the multiplier is the sample and hold detector as presented in [1]. The capacitor voltage is sampled at the rate of the carrier frequency T_0. The phase shift can easily be realized a delay τ of the sampling moment.



Adaptive Sampling Time procedure

The adaptive sampling time demodulation is processed in three steps:

- Detection of the antenna impedance phase shift φ.
- Calculation of the sampling delay time τ
- Setting the delay time of the sampling demodulator

Usually, the phase difference φ is detected by a counter which is clocked by the system clock T_s. In this report a system clock frequency which is 64 times higher than the carrier frequency (T0=64Ts) is employed . Therefore, the phase shift φ is detected in steps of $2\pi/64$. This quantization of the phase shift shall be expressed by:

$$\begin{aligned} \varphi_q &= \left\lfloor \frac{2\cdot\pi}{64}\cdot\varphi\cdot\frac{64}{2\cdot\pi} \right\rfloor \\ &= \frac{2\cdot\pi}{64}\cdot n_\varphi \qquad \text{with } n_\varphi \in 0 \ldots 63 \end{aligned} \tag{22}$$

The “floor” operator $\lfloor x \rfloor$ denotes the next greatest integer less than or equal to x.

Inserting this quantization phase into equation (21), the phase φ_s needs to be adjusted to:

$$\begin{aligned} \varphi_s &= 2\varphi_q - \frac{\pi}{2} \\ &= \frac{2\cdot\pi}{64}\cdot 2\cdot n_\varphi - \frac{\pi}{2} \\ &= \frac{\pi}{32}\cdot 2\cdot n_\varphi - \frac{\pi}{2} \end{aligned} \tag{23}$$

This discrete phase adjustment is implemented in the simulation program. Figure 30 shows both, the continuous and discrete phase adjustment. The sections of the phase steps are easy to recognize. The discrete phase adjustment contour lines are always above the continuous result.

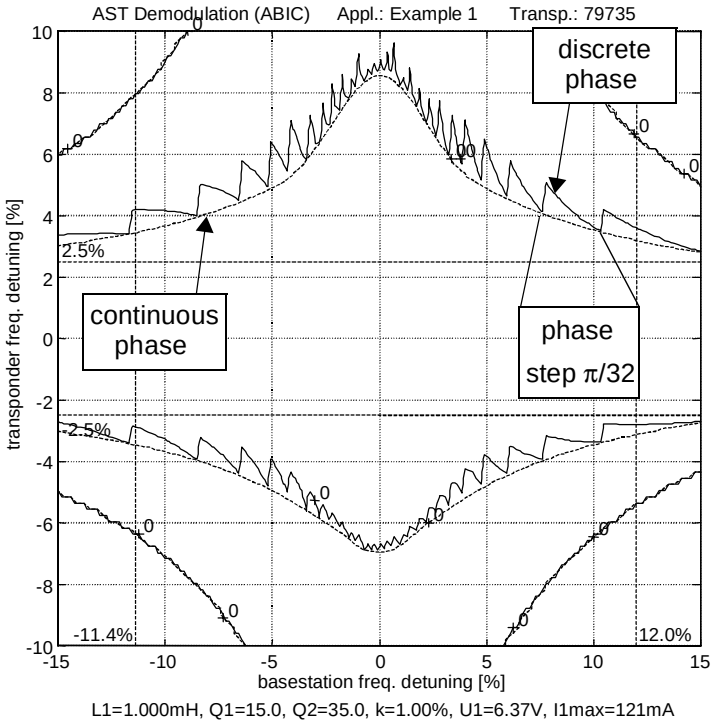


Fig.30 Continuously and discrete phase adjustment (round down function)

This behaviour is caused by floor function of the phase detectior (see equating (22)). A better round function can easily realized by adding 0.5 to phase index n_φ:

$$\varphi_q = \frac{2 \cdot \pi}{64} \cdot (n_\varphi + 0.5)$$

Thus, the phase φ_s needs to be adjusted to:

$$\begin{aligned} \varphi_s &= \frac{\pi}{32} \cdot (2 \cdot (n_\varphi + 0.5)) - \frac{\pi}{2} \\ &= \frac{\pi}{32} \cdot (2 \cdot n_\varphi + 1) - \frac{\pi}{2} \end{aligned} \qquad (24)$$

The simulation results in figure 31 display a symmetrical deviations between the discrete and the continue phase adjustment.

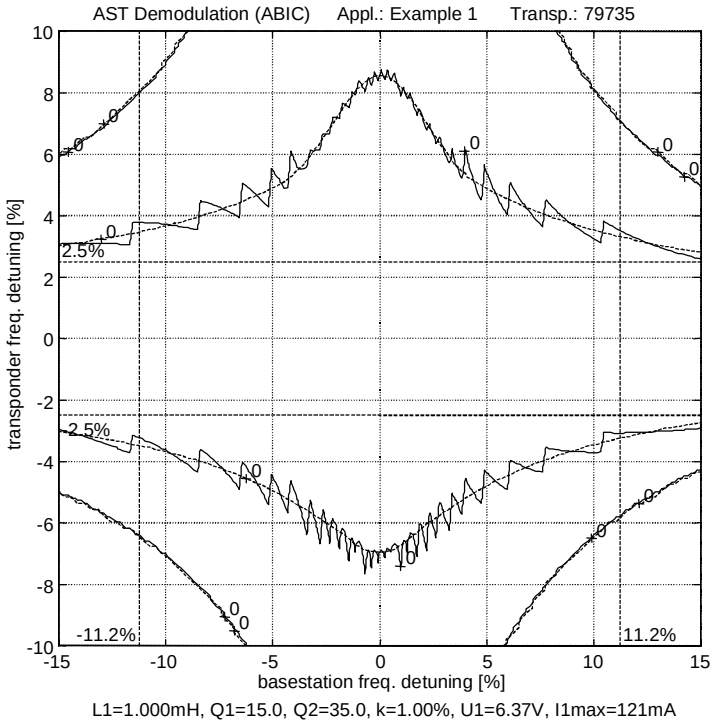


Fig.31 Continuously and discrete phase adjustment (symmetrical round function)

The sampling delay is not used for the simulation model. But, the following equations support the implementation of the AST procedure. The adaptive sampling time can be calculated by:

$$-\omega_0 \cdot \tau = \varphi_s$$

$$\frac{\tau}{T_0} = \frac{\frac{\pi}{2} - \frac{\pi}{32} \cdot (2 \cdot n_\varphi + 1)}{2 \cdot \pi}$$

$$= \frac{1}{4} - \frac{1}{64} \cdot (2 \cdot n_\varphi + 1)$$

Inserting the relation between carrier and system clock $T_0 = 64\ T_s$, the delay can be adjusted in steps of T_S

$$\frac{\tau}{T_s} = 64 \cdot \left(\frac{1}{4} - \frac{1}{64} \cdot (2 \cdot n_\varphi + 1) \right) \quad (25)$$

$$= 16 - (2 \cdot n_\varphi + 1)$$

Using PCF7991 (ABIC)

The PCF7991 Advanced Basestation IC (ABIC) supports this AST procedure. Due to the hardware realization of the ABIC, the phase detector measures the phase between the antenna input voltage and the antenna capacitor voltage. This phase φ_c is shifted by $\pi/2$ to relevant antenna impedance phase φ. Hence the phase φ can be simply calculated.

$$\varphi = \varphi_c - \frac{\pi}{2}$$

The adaptive sampling delay time needs to be externally calculated according the equation (25). A more detailed description how to use the PCF7991 is given in [4].

Another aspect that needs to be considered is the demodulation sensitivity. The demodulation sensitivity is specified in the data sheet [2], but this value can´t employed directly for the simulation. The maximum amplitude of the antenna capacitor voltage is reached when the antenna is tuned exactly.

$$U_{C1max} = Q_1 \cdot U_1 \tag{26}$$

Usually, this value exceeds the maximum demodulator input voltage $U_{in,max}$. Thus, the capacitor voltage needs to be adapted by a voltage divider.

$$\Delta v_{Real} = k_{demod} \cdot Real\left\{\Delta U_{C1} \cdot e^{j\left(2\varphi - \frac{\pi}{2}\right)}\right\}$$

The simulation uses the maximum allowed divider coefficient:

$$k_{demod} = \frac{U_{in,max}}{U_{C1max}} = \frac{U_{in,max}}{Q_1 \cdot U_1} \tag{27}$$

This coefficient is employed for the simulation of the AST demodulator with the parameters given in the data sheet [2]:

- Demodulator input range Demo_Max: Uin,max = 8V
- Receiver sensitivity RXsens: 2mV

Note: The simulation of the ABIC needs to employ a supply voltage of $U_1 = 4/\pi * U_{bat}$ (U_1 = 6.36 V) and an antenna current below the maximum specified value (see [2]).

3. MATLAB® implementation example

3.1 Introduction

The tool MATLAB® (version 4 and 5) was chosen as platform for implementing the simulation program. MATLAB® is based on matrix operations of complex numbers, which is ideal for modelling electronic systems. The tool has extensive graphical visualization capabilities and is available for a large variety of platforms: Unix, Ms-Dos, Windows, etc.

The MATLAB® files contains typical transponder and basestation parameters. For worst case analysis these parameters has to be replace by the minima and maxima values in all combinations.

3.2 Program description

The simulation block diagram is shown in figure 32.

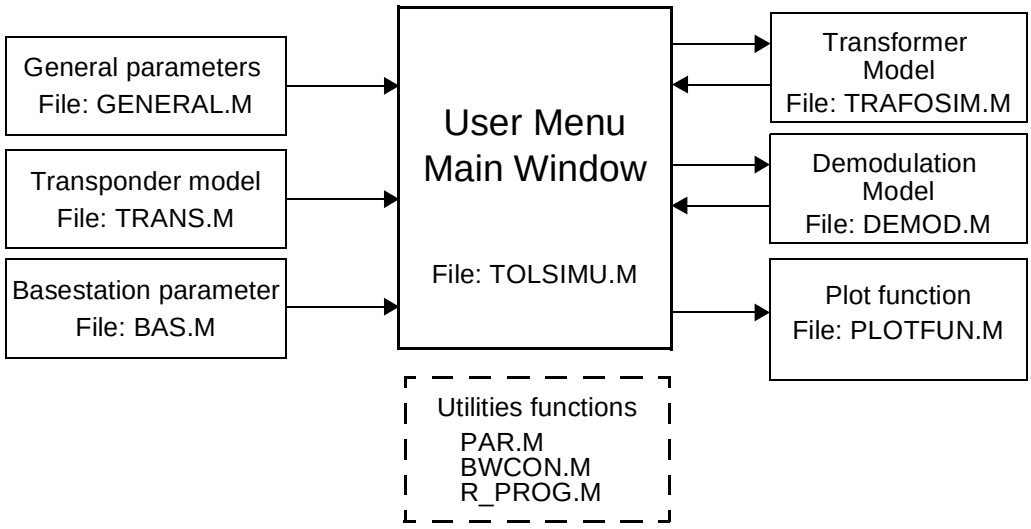
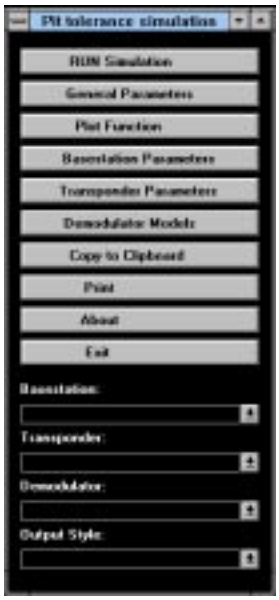


Fig.32 MATLAB® simulation block diagram

The program package contains the following files:

TOLSIMU.M	user menu and control file
TRAFOSYS.M	transformer simulation
BAS.M	basestation parameters
TRANS.M	transponder parameters
DEMOD.M	demodulator types
PLOTFUN.M	plot functions
R_PROG.M	calculation of transponder resistance
BWCON.M	converting plot into printable black&white
PAR.M	function for parallel operator

After the simulation software has been started by typing “tolsimu” in the MATLAB® command window, the following window will be opened:



MATLAB® (version 4)

MATLAB® (version 5)

The push-buttons and four pop-up windows have the following functions:

RUN Simulation: Calculates the simulation results for the selected Application, Demodulator and Transponder, opens an output window and displays the simulation result as specified by the Output Style selector.

General Parameters: Starts the Windows Notepad with the file *general.m* for editing or adding general simulation parameters.

Plot Functions: Starts the Windows Notepad with the file *bas.m* for editing or adding basestation parameters.

Basestation Parameters: Starts the Windows Notepad with the file *bas.m* for editing or adding basestation parameters.

Transponder Parameters: Starts the Windows Notepad with the file *trans.m* for editing or adding transponder parameters.

Demodulator Models: Starts the Windows Notepad with the file *pittoldm.m* for editing or adding demodulator models.

Change Color Mode: Converts the output coloured plot into black&white plot and vice versa.
(Matlab version 5 only)

Copy to Clipboard: Converts the output plot into black&white and copies it to the Windows Clipboard.
(Matlab version 4 only)

Print: Converts into black&white and print the output plot.
(Matlab version 4 only))

Exit: Exits the simulation and closes MATLAB®.

Application: Popup window to select the application parameter set for the simulation. The following choices are included by default:

- Example 1, medium coupling system
- Example 2, low coupling system
- Example 3, high coupling system

Transponder: Popup window to select the transponder parameter set. Following transponders are defined by default:

- 7930XP, read/write transponder
- 79735, SECT security transponder
- 79736, HITAG2 security transponder

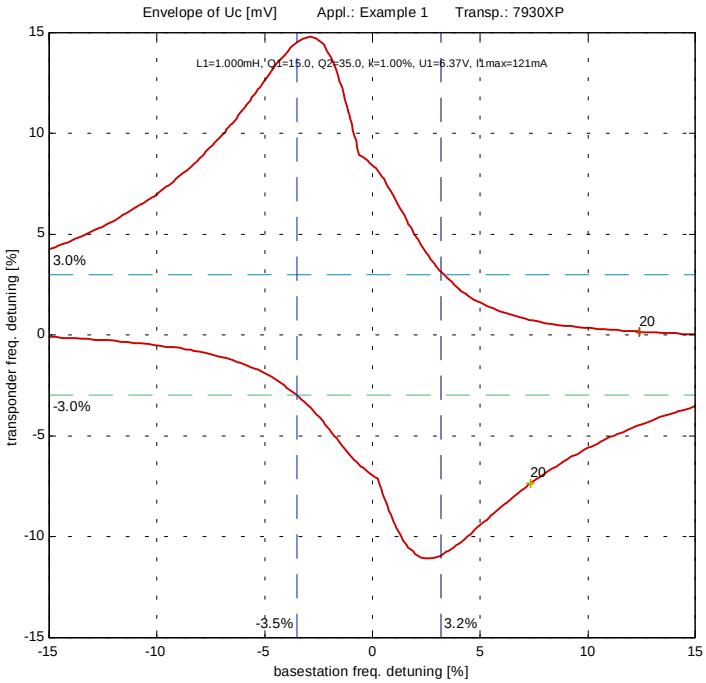
Demodulator: Popup window to select the demodulation model. The following models are selectable by default:

- Envelope of U_c
- AST (ABIC)
- Phase of U_c
- Real Part of Quadrature
- Imaginary Part of Quadrature
- Quadrature, absolute value
- Quadrature, Max(Re, Im)
- Voltage at R_{prog}

Output Style: Popup window to select the display style of the output plot:

- Contour
- Add Contour
- Shaded Contour
- 3D Mesh grid

For example, after choosing “Example 1” from the Application menu, “7930XP” from the transponder menu, “Envelope of Uc mV” from the Demodulator menu, “Contour” from the Output Style menu and finally pressing the “RUN Simulation” button, a window containing the following plot will appear (The plot has been included in this document by using the “Edit / Copy Figure” menu of the Figure “Pit Tolerance Simulation” and choosing Edit/ Paste in the word-processing program):



3.3 Simulation parameter list

TABLE 1 General simulation parameter

Parameter	Symbol	MATLAB® Variable	MATLAB® File	Typical value	Remark
Antenna tolerance range		tol1	GENERAL.M	0.1	
Antenna tolerance simulation steps		step1	GENERAL.M	100	Must be an even number
Transponder tolerance range		tol2	GENERAL.M	0.1	
Transponder tolerance simulation steps		step2	GENERAL.M	100	Must be an even number

TABLE 2 Demodulation parameter

Parameter	Symbol	MATLAB® Variable	MATLAB® File	Typical value	Reference
AST phase offset		DeltaN	DEMOD.M	1	Delta controls type of the round function: DeltaN=0: floor function DeltaN=1: round function DeltaN=2: round up function See chapter 2.5.4

TABLE 3 Basestation parameter

Parameter	Symbol	MATLAB® Variable	MATLAB® File	Typical value	Reference
Source voltage	U_1	U1	BAS.M		chapter 2.1
Antenna inductance	L_1	L1	BAS.M		chapter 2.1
Antenna quality	Q_1	Q1	BAS.M		chapter 2.1
Coupling coefficient	k	k	BAS.M		chapter 2.1
Demodulator sensitivity		demodsense	BAS.M DEMOD.M		chapter 2.5.1

TABLE 4 Transponder parameter

Parameter	Symbol	MATLAB® Variable	MATLAB® File	Typical value for PCF79735	Typical value for PCF79736	Reference
Transponder inductance	L_2	L2	TRANS.M	2.33mH	2.33mH	chapter 2.1
Transponder quality	Q_2	Q2	TRANS.M	35	35	chapter 2.1
Programming voltage	U_{prog}	Uprog	TRANS.M	9 V	4 V	chapter 2.4
Upper clipping voltage	U_{clip1}	Uclip1	TRANS.M	9 V	6 V	chapter 2.4
Upper basic load resistance	R_{bas1}	Rmax	TRANS.M	200 kΩ	200 kΩ	chapter 2.4
Lower clipping voltage A	U_{clip2a}	Uclip2a	TRANS.M	3 V	6 V	chapter 2.4
Lower basic load resistance A	R_{bas2a}	RA	TRANS.M	200 kΩ	4.7 kΩ	chapter 2.4
Lower clipping voltage B	U_{clip2b}	Uclip2a	TRANS.M	3 V	2 V	chapter 2.4
Lower basic load resistance B	R_{bas2b}	RA	TRANS.M	200 kΩ	2.2 kΩ	chapter 2.4
Flux density for Program_Mode	B_{prog}	Bprog	TRANS.M	100 µT	35 µT	chapter 2.3
Flux density coefficient	γ	gamma	TRANS.M	20mV/µT	20mV/µT	chapter 2.3

4. Measurement of system parameters

In order to verify the actual system against the simulation results, the following characteristic system parameters need to be known:

- Coupling coefficient between basestation and transponder antenna: k
- Basestation antenna parameters: L_1, Q_1
- Basestation output driver voltage: U_1
- Transponder parameters: L_2, Q_2, resonant frequency tolerances, clipping levels U_{clip1}, U_{clip2}, programming flux density $B_{prog.}$
- To verify a specific operating point, actual resonant frequencies of basestation and transponder antennas: f_{res1}, f_{res2}.

The transponder parameters are defined in the corresponding specifications; typical values for $U_{clip1,2}$ are inserted in the simulation models. Measurement methods for the most important system and basestation related parameters are introduced in the following.

4.1 Coupling coefficient

In order to measure the system´s coupling coefficient, Philips provides modified transponder packages (“moulded coils”), which have the integrated circuit and the resonant capacitor removed, and the coil connectors accessible for external probing. The basestation antenna needs to be connected to a sine wave generator, adjusted to the transponder´s operating frequency of 125 kHz. A properly adjusted low-capacitance oscilloscope probe should be used to measure the voltage U_2 as shown in figure 33. Proper probe adjustment is important, since the transponder coil has a relatively high impedance which makes it essential to measure current less (I_2=0).

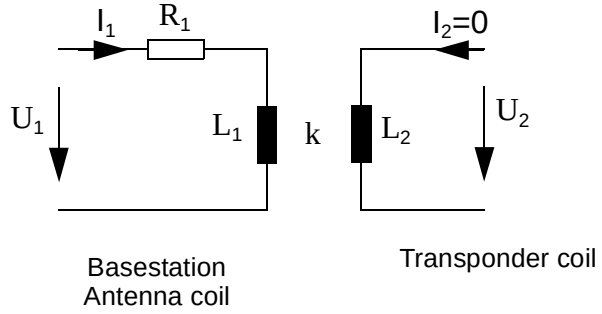


Fig.33 Measurement circuit for k

The coupling coefficient can be calculated as follows:

$$U_1 = I_1 (R_1 + j\omega L_1)$$

$$U_2 = I_1 \cdot j\omega M$$

with

$$M = k \cdot \sqrt{L_1 L_2}$$

follows

$$\frac{U_2}{U_1} = \frac{j\omega M \cdot I_1}{(R_1 + j\omega L_1) \cdot I_1} = \frac{j\omega \cdot k \cdot \sqrt{L_1 L_2}}{R_1 + j\omega L_1}$$

and

$$k = \left|\frac{U_2}{U_1}\right| \cdot (R_1 + j\omega L_1) \cdot \frac{1}{j\omega\sqrt{L_1 L_2}}$$

Provided the quality factor is higher than approx. 10, the calculation can be simplified with less than 1% error:

$$Q_1 = \frac{\omega L_1}{R_1} > 10$$

$$R_1 + j\omega L_1 \approx j\omega L_1$$

$$k \approx \left|\frac{U_2}{U_1}\right| \cdot \sqrt{\frac{L_1}{L_2}} \qquad (28)$$

4.2 Basestation antenna quality

For worst case considerations, the basestation antenna circuit needs to be evaluated extensively regarding the overall antenna quality Q_1. Figure 34 shows a more in-depth circuit diagram of the antenna circuit.

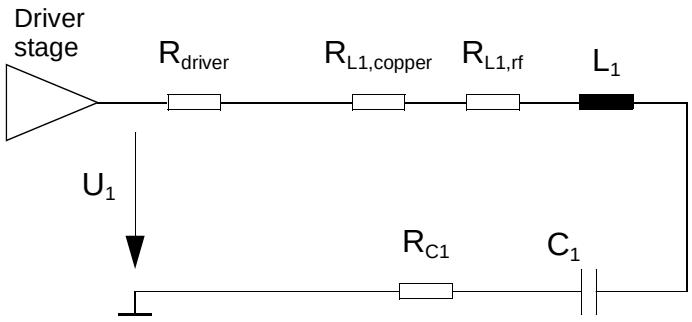


Fig.34 Detailed equivalent circuit of basestation antenna

The basestation antenna quality Q_1 is influenced by any of the resistive components, which leads to following expression:

$$Q_1 = \frac{\omega L_1}{R_1}$$

$$R_1 = R_{driver} + R_{L1,copper} + R_{L1,rf} + R_{C1}$$

The single resistive components to be measured are:

- R_{driver} The output impedance of the basestation driver stage.
- $R_{L1,copper}$ The resistance of the copper wire that form the antenna coil, which needs to be investigated especially under worst case temperature conditions.
- $R_{L1,rf}$ The resistive part of the eddy current losses that may result from metallic parts in the vicinity of the antenna coil, for example a car lock barrel which the coil is mounted on.
- R_{C1} The resistive part of the capacitor, which is negligible in most cases.

Other resistive parts might be considered, for example long antenna leads or connectors.

Notice that the voltage U_1 denotes the open circuit voltage of the driver stage.

4.3 Transponder resonant frequency

The resonant frequency of an actual transponder needs to be measured contactless. The measurement setup in figure 35 comprises an impedance analyser or similar equipment, which supplies a measurement antenna coil with an RF voltage of variable frequency. A close coupling between transponder and measurement coil is recommended for accurate results.

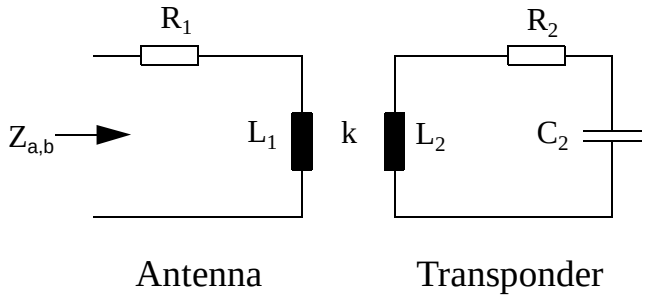


Fig.35 Contactless measurement of transponder resonant frequency

The generator amplitude should be chosen such that the transponder circuit will not operate and stays high ohmic. The measurement method arises directly from the following relations:

The impedances of the system including or not including the transponder are:

$$Z_a = R_1 + j\omega L_1 + \frac{(\omega M)^2}{R_2 + j\omega L_2 + \frac{1}{j\omega C_2}}$$

$$Z_b = R_1 + j\omega L_1$$

The differential impedance can be calculated:

$$\Delta Z = Z_a - Z_b$$

$$\Delta Z = \frac{(\omega M)^2}{R_2 + j\omega L_2 + \frac{1}{j\omega C_2}}$$

The phase of ΔZ results as

$$\arg(\Delta Z) = -\arctan\left(\frac{\omega L_2 - \frac{1}{\omega C_2}}{R_2}\right)$$

The phase difference will only be zero when the generator´s angular frequency w equals the resonant angular frequency of the transponder:

$$\arg(\Delta Z) = 0 \Rightarrow \omega = \frac{1}{\sqrt{L_2 C_2}} = \omega_{r2}$$

The measurement method comprises the following iterative steps:

1. Insert the transponder in the RF field of the measurement coil. Measure the resulting impedance Z_a of the setup at the antenna.
2. Remove the transponder from the measurement coil. Measure the antenna impedance Z_b.
3. Check whether $\arg(Z_a - Z_b) = 0$, vary the generator frequency and go back to step 1 if not.
4. The current generator frequency equals the resonant frequency of the examined transponder.

5. Reference List

[1] Misha Schwartz: Information Transmission, Modulation and Noise
Mc Graw Hill 1970

[2] Data Sheet : PCF7991AT, Advanced Basestation IC

[3] Data Sheet: PCF79736, Security Transponder

[4] Application Note: Designing RF-identification basestations with the Advanced Basestation IC PCF7991
Philips Semiconductors.

Note: For data sheets and application notes please contact your local sales organization.

APPENDIX 1 Open circuit voltage and output impedance

Open circuit voltage

Using the equivalent transformer circuit, the following equation for the open circuit voltage function can be derived:

$$\frac{U_0}{U_1} = \frac{XC_2}{R_2 + XL_{s2} + XC_2} \cdot \frac{\text{par}(XM, (R_2 + XL_{s2} + XC_2))}{R_1 + XL_{s1} + XC_1 + \text{par}(XM, (R_2 + XL_{s2} + XC_2))}$$

With the definitions

$$\Delta X_1 = XL_1 + XC_1 = XL_{s1} + XC_1 + XM \qquad \Delta X_2 = XL_2 + XC_2 = XL_{s2} + XC_2 + XM$$

the voltage transfer function can be simplified:

$$\frac{U_0}{U_1} = \frac{XC_2}{R_2 + XL_{s2} + XC_2} \cdot \frac{\frac{XM(R_2 + \Delta X_2 - XM)}{R_2 + \Delta X_2}}{R_1 + \Delta X_1 - XM + \frac{XM(R_2 + \Delta X_2 - XM)}{R_2 + \Delta X_2}}$$

$$\frac{U_0}{U_1} = \frac{XC_2}{R_2 + XL_{s2} + XC_2} \cdot \frac{XM(R_2 + \Delta X_2 - XM)}{(R_1 + \Delta X_1 - XM)(R_2 + \Delta X_2) + XM(R_2 + \Delta X_2 - XM)}$$

$$\frac{U_0}{U_1} = \frac{XC_2 \cdot XM}{(R_1 + \Delta X_1) \cdot (R_2 + \Delta X_2) - XM^2}$$

Introducing the resonant condition for both resonant circuits

$$XC_1 = -XL_1 \cdot \frac{\omega_{r1}^2}{\omega_0^2} \qquad XC_2 = -XL_2 \cdot \frac{\omega_{r2}^2}{\omega_0^2}$$

where ω_{r1} **denotes the resonant frequency of the basestation,** ω_{r2} **the resonant frequency of the transponder and** ω_0 **the carrier frequency,** Δx_1 **and** Δx_2 **can be written as**

$$\Delta X_1 = XL_1\left(1 - \frac{\omega_{r1}^2}{\omega_0^2}\right) \qquad \Delta X_2 = XL_2\left(1 - \frac{\omega_{r2}^2}{\omega_0^2}\right)$$

From that, tolerance coefficients can be defined:

$$p_1 = 1 - \left(\frac{\omega_{r1}}{\omega_0}\right)^2 \qquad p_2 = 1 - \left(\frac{\omega_{r2}}{\omega_0}\right)^2$$

or

$$\frac{\omega_{r1}^2}{\omega_0^2} = 1 - p_1 \qquad \frac{\omega_{r2}^2}{\omega_0^2} = 1 - p_2$$

With substitution of:

$$XC_2 = -(1 - p_2)XL_2 \qquad XC_1 = -(1 - p_2)XL_1$$

$$\Delta x_1 = p_1 \cdot XL_1 \qquad \Delta x_2 = p_2 \cdot XL_2$$

$$XM^2 = k^2 \cdot XL_1 \cdot XL_2$$

the open circuit voltage function can be written as

$$\frac{U_0}{U_1} = \frac{(p_2 - 1) \cdot XL_2 \cdot \sqrt{XL_2 \cdot XL_1} \cdot k}{(R_1 + p_1 XL_1)(R_2 + p_2 XL_2) - \left(k^2 \cdot XL_1 \cdot XL_2\right)}$$

and simplified:

$$\frac{U_0}{U_1} = \frac{(p_2 - 1) \cdot \sqrt{\frac{XL_2}{XL_1}} \cdot k}{\left(\frac{R_1}{XL_1} + p_1\right)\left(\frac{R_2}{XL_2} + p_2\right) - k^2}$$

With definitions of antenna quality factors

$$jQ_1 = \frac{XL_1}{R_1} \qquad jQ_2 = \frac{XL_2}{R_2}$$

the equation can be written as

$$\frac{u_0}{U_1} = \frac{(1 - p_2) \cdot \sqrt{\frac{XL_2}{XL_1}} \cdot k}{k - \left(\frac{1}{jQ_1} + p_1\right)\left(\frac{1}{jQ_2} + p_2\right)}$$

and results as

$$\frac{U_0}{U_1} = \frac{Q_1 \cdot Q_2 \cdot k \cdot (1 - p_2) \cdot \sqrt{\frac{L_2}{L_1}}}{Q_1 \cdot Q_2 \cdot k^2 + (1 + jQ_1 \cdot p_1)(1 + jQ_2 \cdot p_2)}$$

Output impedance

From the equivalent transformer circuit, the following equation for the output impedance can be derived:

$$Z_i = \mathrm{par}(XC_2, (XL_{s2} + R_2 + \mathrm{par}(XL_{s1} + R_1 + XC_1, XM)))$$

After expanding the parallel operators,

$$Z_i = \frac{XC_2 \cdot \left(R_2 + XL_{s2} + \frac{XM \cdot (R_1 + XL_{s1} + XC_1)}{XM + R_1 + XL_{s1} + XC_1} \right)}{R_2 + XL_{s2} + XC_2 + \frac{XM \cdot (R_1 + XL_{s1} + XC_1)}{XM + R_1 + XL_{s1} + XC_1}}$$

With the definitions of ΔX_1 and ΔX_2, the output impedance can be simplified:

$$Z_i = XC_2 \cdot \frac{(R_2 + XL_2 - XM) \cdot (R_1 + \Delta X_1) + XM \cdot (R_1 + \Delta X_1 - XM)}{(R_2 + \Delta X_2 - XM) \cdot (R_1 + \Delta X_1) + XM \cdot (R_1 + \Delta X_1 - XM)}$$

$$Z_i = XC_2 \cdot \frac{(R_2 + XL_2) \cdot (R_1 + \Delta X_1) - XM(R_1 + \Delta X_1) + XM \cdot (R_1 + \Delta X_1) - XM^2}{(R_2 + \Delta X_2) \cdot (R_1 + \Delta X_1) - XM(R_1 + \Delta X_1) + XM \cdot (R_1 + \Delta X_1) - XM^2}$$

$$Z_i = XC_2 \cdot \frac{(R_2 + XL_2) \cdot (R_1 + \Delta X_1) - XM^2}{(R_2 + \Delta X_2) \cdot (R_1 + \Delta X_1) - XM^2}$$

The equation can be expressed by the quality definition Q_1, Q_2 and the tolerance coefficients p1,p2:

$$Z_i = XC_2 \cdot \frac{\left(1 + \frac{XL_2}{R_2}\right) \cdot \left(1 + \frac{\Delta X_1}{R_1}\right) - \frac{XM^2}{R_1 \cdot R_2}}{\left(1 + \frac{\Delta X_2}{R_2}\right) \cdot \left(1 + \frac{\Delta X_1}{R_1}\right) - \frac{XM^2}{R_1 \cdot R_2}}$$

$$Z_i = -(1 - p_2) \cdot XL_2 \cdot \frac{(1 + jQ_1 \cdot p_1) \cdot (1 + jQ_2) + Q_1 \cdot Q_2 \cdot k^2}{(1 + jQ_1 \cdot p_1) \cdot (1 + jQ_2 \cdot p_2) + Q_1 \cdot Q_2 \cdot k^2}$$

$$Z_i = -(1 - p_2) \cdot XL_2 \cdot \frac{(1 + jQ_1 \cdot p_1) \cdot (1 + jQ_2 + jQ_2 \cdot p_2 - jQ_2 \cdot p_2) + Q_1 \cdot Q_2 \cdot k^2}{(1 + jQ_1 \cdot p_1) \cdot (1 + jQ_2 \cdot p_2) + Q_1 \cdot Q_2 \cdot k^2}$$

$$Z_i = -(1 - p_2) \cdot XL_2 \cdot \left(\frac{jQ_2 \cdot (1 + jQ_1 \cdot p_1) \cdot (1 - p_2)}{(1 + jQ_1 \cdot p_1) \cdot (1 + jQ_2 \cdot p_2) + Q_1 \cdot Q_2 \cdot k^2} + 1 \right)$$

$$Z_i = \omega_0 L_2 \cdot (1 - p_2) \cdot \left(\frac{Q_2 \cdot (1 + jQ_1 \cdot p_1)(1 - p_2)}{(1 + jQ_1 \cdot p_1) \cdot (1 + jQ_2 \cdot p_2) + Q_1 \cdot Q_2 \cdot k^2} - j \right)$$

APPENDIX 2 Antenna input impedance

The antenna input impedance depending of the load resistance can be simplified, starting with equation

$$Z_{g1,2} = R_1 + XL_{s1} + XC_1 + \mathrm{par}(XM, R_2 + XL_{s2} + \mathrm{par}(XC_2, R_{load1,2}))$$

First, the parallel operator is expanded:

$$Z_{g1,2} = R_1 + XL_{s1} + XC_1 + \frac{XM \cdot \left(R_2 + XL_{s2} + \frac{XC_2 \cdot R_{load1,2}}{XC_2 + R_{load1,2}}\right)}{XM + R_2 + XL_{s2} + \frac{XC_2 \cdot R_{load1,2}}{XC_2 + R_{load1,2}}}$$

With the definitions of ΔX_1 and ΔX_2, the equation can be simplified to

$$Z_{g1,2} = R_1 + \Delta X_1 - XM + \frac{XM \cdot ((R_2 + \Delta X_2 - XM) \cdot (XC_2 + R_{load1,2}) + XC_2 \cdot R_{load1,2})}{XM \cdot (XC_2 + R_{load1,2}) + (R_2 + \Delta X_2 - XM) \cdot (XC_2 + R_{load1,2}) + XC_2 \cdot R_{load1,2}}$$

$$Z_{g1,2} = R_1 + \Delta X_1 + XM \cdot \left(\frac{(R_2 + \Delta X_2 - XM) \cdot (XC_2 + R_{load1,2}) + XC_2 \cdot R_{load1,2}}{XM \cdot (XC_2 + R_{load1,2}) + (R_2 + \Delta X_2 - XM) \cdot (XC_2 + R_{load1,2}) + XC_2 \cdot R_{load1,2}} - 1\right)$$

$$Z_{g1,2} = R_1 + \Delta X_1 - \left(\frac{XM^2}{R_2 + \Delta X_2 + \frac{XC_2 \cdot R_{load1,2}}{(XC_2 + R_{load1,2})}}\right)$$

The dependence of system parameter Q_1, Q_2 and the tolerance coefficients p1,p2 is given by:

$$Z_{g1,2} = R_1 \cdot \left(1 + \frac{\Delta X_1}{R_1} - \left(\frac{\frac{k^2 \cdot XL_1 \cdot XL_2}{R_1 \cdot R_2}}{1 + \frac{\Delta X_2}{R_2} + \frac{\Delta X_2 - XL_2}{R_2} \cdot \frac{R_{load1,2}}{\Delta X_2 - XL_2 + R_{load1,2}}}\right)\right)$$

$$Z_{g1,2} = R_1 \cdot \left(1 + jQ_1 \cdot p_1 + \left(\frac{Q_1 \cdot Q_2 \cdot k^2}{1 + jQ_2 \cdot p_2 + Q_2 \cdot \frac{R_{load1,2} \cdot (1 - p_2)}{\omega_0 L_2 \cdot (1 - p_2) + j \cdot R_{load1,2}}}\right)\right)$$

APPENDIX 3 MATLAB® listing: Main program (tolsimu.m)

```
%|-------------------------------------------------------|
%|         Philips Semiconductors, SLH                   |
%|-------------------------------------------------------|
%| function tolsimu(action)                              |
%|                                                       |
%| matlab-application for calculation of the tolerances  |
%| of systems with inductive coupled transponders        |
%|                                                       |
%| Version: 3.0                                          |
%|                                                       |
%|-------------------------------------------------------|
%| File: TOLSIMU.M                                       |
%| Author(s): Frank Boeh                                 |
%|            Wolfgang Tobergte                          |
%|                                                       |
%| Upgrade for Matlab 5.2                                |
%|                                                       |
%|-------------------------------------------------------|
function tolsimu(action);
if exist('globdef')==2
   globdef;
end
pic=2; %standard simulation figure
% MATLAB version detection
if findstr(version,'5.')==1,
   mversion=5;
   deltatext=25;
   deltamainhigh=-20;
   editstr='!notepad ';
else
   mversion=4;
   deltatext=20;
   deltamainhigh=0;
   editstr='!notepad ';
end
if nargin<1,   % ... for first call of pittol
   action='initialize';
end;
%-------------------------------------------------------%
% Case 1: First call of tolsimu : Initialize            %
%                                                       %
% actions:                                              %
% --------                                              %
% * make main window                                    %
% * get parameters for popup-windows                    %
%                                                       %
%-------------------------------------------------------%
if strcmp(action,'initialize'),
set(0,'visible','off');
figure(1);
close(1);
mainposy = 0;  % position of main window
mainposx = 0;
mainwidth = 250;
mainheight = 480+40+ deltamainhigh;
main=figure(1);
if mversion==5,
   temp='off';
else
   temp='no';
end
```

```
set(main,...
    'Visible','off',...
    'ShareColors',temp,...
    'Resize','off',...
    'Units','pixels',...
    'NumberTitle','off',...
    'Position',[mainposx mainposy mainwidth mainheight],...
    'Menu','None',...
    'Color',[0 0.5 0.75],...
    'Name','Pit tolerance simulation');
if mversion==5,
   set(main,...
      'ShareColors','off',...
      'CloseRequestFcn','tolsimu(''exit    '');');
end
set(main,'units','norm');
scpos=get(main,'pos');
scpos(2)=1-scpos(4)-0.1;
scpos(1)=0.05;
set(main,'pos',scpos);
set(main,'units','pixel');
set(gca,'visible','off');
if findstr(version,'5.')==1,
    label = str2mat(...   % define buttons
    'RUN Simulation', ...
    'General Parameters', ...
    'Plot Function', ...
    'Basestation Parameters', ...
    'Transponder Parameters', ...
    'Demodulator Models', ...
    'Change Color Mode', ...
    'About');
   callbacks = [ ...                % callbacks:
    'tolsimu(''run     '');'     % ... for RUN Simulation
    'tolsimu(''paraedit'');'     % ... for Display parameters
    'tolsimu(''dispedit'');'     % ... for Display parameters
    'tolsimu(''baseedit'');'     % ... for Basestation parameters
    'tolsimu(''trnsedit'');'     % ... for Transponder parameters
    'tolsimu(''dmedit  '');'     % ... for Demodulator parameters
    'tolsimu(''chcolm  '');'     % ... for Demodulator parameters
    'tolsimu(''help    '');'];    % ... for help
else
  label = str2mat(...   % define buttons
    'RUN Simulation', ...
    'General Parameters', ...
    'Plot Function', ...
    'Basestation Parameters', ...
    'Transponder Parameters', ...
    'Demodulator Models', ...
    'Copy to Clipboard', ...
    'Print', ...
    'About',...
    'Exit');
   callbacks = [ ...                % callbacks:
    'tolsimu(''run     '');'     % ... for RUN Simulation
    'tolsimu(''paraedit'');'     % ... for Display parameters
    'tolsimu(''dispedit'');'     % ... for Display parameters
    'tolsimu(''baseedit'');'     % ... for Basestation parameters
    'tolsimu(''trnsedit'');'     % ... for Transponder parameters
    'tolsimu(''dmedit  '');'     % ... for Demodulator parameters
    'tolsimu(''copy    '');'     % ... for Copy to Clipboard
    'tolsimu(''print   '');'     % ... for Copy to Clipboard
    'tolsimu(''help    '');'     % ... for help
```

```
    'tolsimu(''exit    '');'];   % ... for exit
end
buttonwidth = mainwidth - 20;   % define size of buttons
buttonheight = 25;
buttonposx = (mainwidth-buttonwidth)/2;
buttonposy = mainheight - 40;

for i=1:size(label,1)   % make buttons
   toluic(i)=uicontrol('Style','Pushbutton',...
                       'Units','Pixels',...
                       'Position',[buttonposx buttonposy buttonwidth buttonheight],...
                       'String',label(i,:),...
                       'Callback',callbacks(i,:));
   buttonposy = buttonposy - 30;  % position-offset for next button
end
typ=1;
bas;     % get basestation-parameters
trans;   % get Transponder- list
demod;   % get demodulator-list
plotfun; % get plot function list
  typ = 1; % default basestation-type
  transpondertyp = 1; % default transponder-type
  demodtyp = 1;% default demodulator-type
  plottyp=1;
  dreid = 1; % default output-plot
fontsize=10;
textwidth = 12*fontsize; % define size of popup-menus
textheight = fontsize+6;
textposx = buttonposx;
textposy = buttonposy+5+(5-mversion)*3;
apptext=uicontrol('Style','text',... % label for basestation-menu
       'Units','pixel',...
       'Backgroundcolor',[0 0.5 0.75],...
       'Position',[textposx textposy textwidth textheight],...
       'Foregroundcolor',[1 1 1],...
       'String','Basestation:                    ',...
       'HorizontalAlignment','left');
if mversion==5,
     set(apptext,'Fontsize',fontsize);
end

textposy = textposy - deltatext;
tyuic=uicontrol('Style','Popup',... % make basestation-popup-menu
                'Units','pixels',...
                'Position',[textposx textposy buttonwidth textheight],...
                'Backgroundcolor',[0.8 0.8 0.8],...
                'String',typelist,...
                'Value',typ);
textposy = textposy - 30;
modtext=uicontrol('Style','text',... % label for transponder-menu
                  'Units','pixels',...
                  'Position',[textposx textposy textwidth textheight],...
                  'Backgroundcolor',[0 0.5 0.75],...
                  'Foregroundcolor',[1 1 1],...
                  'String','Transponder:                      ',...
                  'HorizontalAlignment','left');
if mversion==5,
     set(modtext,'Fontsize',fontsize);
end
textposy = textposy - deltatext;
transuic=uicontrol('Style','Popup',... % make transponder-type popup-menu
                   'Units','pixels',...
                   'Position',[textposx textposy buttonwidth textheight],...
```

```
                  'Backgroundcolor',[0.8 0.8 0.8],...
                  'String',translist,...
                  'Value',transpondertyp);
textposy = textposy - 30;
modtext=uicontrol('Style','text',... % label for demodulator-menu
                  'Units','pixels',...
                  'Position',[textposx textposy textwidth textheight],...
                  'Backgroundcolor',[0 0.5 0.75],...
                  'Foregroundcolor',[1 1 1],...
                  'String','Demodulator:                          ',...
                  'HorizontalAlignment','left');

if mversion==5,
     set(modtext,'Fontsize',fontsize);
end
textposy = textposy - deltatext;
moduic=uicontrol('Style','popup',... % make demodulator-popup-menu
              'Units','pixels',...
                 'Position',[textposx textposy buttonwidth textheight],...
                 'Backgroundcolor',[0.8 0.8 0.8],...
                 'String',demodlist,...
                 'value',demodtyp);
textposy = textposy - 30;
modtext=uicontrol('Style','text',... % label for output-style-menu
                  'Units','pixels',...
                  'Position',[textposx textposy textwidth textheight],...
                  'Backgroundcolor',[0 0.5 0.75],...
                  'Foregroundcolor',[1 1 1],...
                  'String','Output Style:                          ',...
                  'HorizontalAlignment','left');
if mversion==5,
     set(modtext,'Fontsize',fontsize);
end
textposy = textposy - deltatext;
dreiuic=uicontrol('Style','Popup',... % make popup-window for output-plot-style
                  'Units','pixels',...
                  'Position',[textposx textposy buttonwidth textheight],...
                  'Backgroundcolor',[0.8 0.8 0.8],...
                  'String',plotlist,...
                  'Value',plottyp);
set(main, 'Visible','on'); % make main-window visible
set(main, 'UserData', [transuic moduic dreiuic tyuic]); % store all relevant data
                                                        % in the user-data of main-window
%--------------------------------------------------------%
% Case 2: call of tolsimu by the run-button             %
%                                                        %
% actions:                                               %
% --------                                               %
% * get parameters for calculation                       %
% * calculate data                                       %
% * make plot-window                                     %
%                                                        %
%--------------------------------------------------------%
elseif strcmp(action,'run     '),
  clear functions
  hndlList = get(1,'UserData');   % get all relevant data from user-data of main-window
  transuic = hndlList(1);    % get all handles
  moduic = hndlList(2);
  dreiuic = hndlList(3);
  tyuic = hndlList(4);
  transpondertyp = get(transuic,'Value');% get the values
  demodtyp = get(moduic,'Value');
  plottyp = get(dreiuic,'Value');
```

```
  typ = get(tyuic,'Value');
  general;        % get general simulatio parameters
%-------------------------------------------------------------------------%
  if (exist('mypara.m')==2),
     mypara;
  end
%-------------------------------------------------------------------------%
  bas;     % get application-parameters
  trans; % get transponder-parameters


  trafosim;       % Kernel simulation
  demod;          % calculate demodulation
  if ~exist('v')
    mindv = min(min(dv));      % get minimum value of dv
    maxdv = max(max(dv));      % get maximum value of dv
    if exist('demodsense')
       v = [0 demodsense linspace(mindv, maxdv, 3)];  % calculate 5 lines for contour-plot
    else
       v = [0 linspace(mindv, maxdv, 4)];  % calculate 5 lines for contour-plot
    end;
  end
  plttext = sprintf('L1=%4.3fmH, Q1=%2.1f, Q2=%2.1f, k=%3.2f%%, U1=%3.2fV, I1max=%3.0fmA',...
                    1e3* L1   , Q1      , Q2      , 100*k    , U1       , i1max);
  plotfun;
  % save plottyp -> UserData of figure 2
  a=get(gcf,'UserData');
  set(gcf,'UserData',[plottyp a(2)]);
  if (mversion==5),
     bwcon;
  end
%-----------------------------------------------------------------%
% Case 3: Call of tolsimu by Gereral parameter-button             %
%                                                                 %
% action:                                                         %
% --------                                                        %
% * start notepad with      bas.m                                 %
%                                                                 %
%-----------------------------------------------------------------%
elseif strcmp(action,'paraedit'),
    if (exist('mypara')==2)
      eval([editstr ' mypara.m'])
    end
%-----------------------------------------------------------------%
% Case 3: Call of tolsimu by Output Style Function -button        %
%                                                                 %
% action:                                                         %
% --------                                                        %
% * start notepad with plotfun.m                                  %
%                                                                 %
%-----------------------------------------------------------------%
elseif strcmp(action,'dispedit'),
    if (exist('myplot')==2)
      eval([editstr ' myplot.m'])
      plotfun  % get Plot function
            h = get(1,'UserData');   % get all relevant data from user-data of main-window
            h = h(3);
      set(h,'String',plotlist);
      clear all
    end
```

```
%-------------------------------------------------------------%
% Case 4: Call of tolsimu by Basestation-parameter-button    %
%                                                             %
% action:                                                     %
% --------                                                   %
% * start notepad with bas.m                                  %
%                                                             %
%-------------------------------------------------------------%
elseif strcmp(action,'baseedit'),
    if exist('myappl')==2
      eval('!notepad myappl.m')
       h = get(1,'UserData');   % get all relevant data from user-data of main-window
       typ = get(h(4),'Value');
      bas; % get basestation-parameters
       h = get(1,'UserData');   % get all relevant data from user-data of main-window
      set(h(4),'String',typelist);
      clear all
  end
%-------------------------------------------------------------%
% Case 5: Call of tolsimu by Transponder-parameter-button  %
%                                                             %
% action:                                                     %
% --------                                                   %
% * start notepad with trans.m                                %
%                                                             %
%-------------------------------------------------------------%
elseif strcmp(action,'trnsedit'),
    if exist('mytrans')==2
       eval('!notepad mytrans.m')
        trans; % get transponder-list
        h = get(1,'UserData');   % get all relevant data from user-data of main-window
        h = h(1);
        set(h,'String',translist);
        clear all
    end

%-------------------------------------------------------------%
% Case 6: Call of tolsimu by Demodulator-parameter-button  %
%                                                             %
% action:                                                     %
% --------                                                   %
% * start notepad with demod.m                                %
%                                                             %
%-------------------------------------------------------------%
elseif strcmp(action,'dmedit  '),
    if exist('mydemod')==2
        eval('!notepad mydemod.m')
        demod; % get demodulator-list
        h = get(1,'UserData');   % get all relevant data from user-data of main-window
        h = h(2);
        set(h,'String',demodlist);
        clear all
    end
%-----------------------------------------------------%
% Case 7: Call of tolsimu by Copy2Clipboard           %
%                                                     %
% action:                                             %
% --------                                           %
% * recall tolsimu(run)                               %
% * convert plot to plain format                      %
% * copy plot to clipboard                            %
% * close figure                                      %
%                                                     %
```

```
%-------------------------------------------------------%
elseif strcmp(action,'copy    '),
   screen = get(0, 'ScreenSize');   % get Screen-Size for plot-window-position
   width = screen(3);
   height = screen(4);
   figposx = 242;     % position of plot-window
   figposy = height;
   if height<700      % size of plot-window
      figheight = height - 10;
   else
      figheight = 700;
   end
   if width<900
      figwidth = width - 250;
   else
      figwidth = 700;
   end
   if (isempty(find(get(0,'child')==2))),
       tolsimu('run     ');
   end
   figpos= get(2,'Position');
   set(2,...% define plot-window
          'Visible','off',...% define plot-window
          'Units','pixels',...
          'NumberTitle','Off',...
          'Menu','None',...
          'Position',[figposx figposy figheight figwidth],...
          'Color',[0.0 0.0 0.0]);
   bwcon;
   figure(2);
   print -dmeta;
   set(2,'Position',figpos,'color','default','Visible','off');
   close(2)
   tolsimu('run     ');
%-------------------------------------------------------%
% Case 8: Call of tolsimu by print-button              %
%                                                      %
% action:                                              %
% --------                                             %
% * print the simulatiom plot                          %
%                                                      %
%-------------------------------------------------------%
elseif strcmp(action,'print   '),

   screen = get(0, 'ScreenSize');   % get Screen-Size for plot-window-position
   width = screen(3);
   height = screen(4);
   figposx = 242;     % position of plot-window
   figposy = height;
   if height<700      % size of plot-window
      figheight = height - 10;
   else
      figheight = 700;
   end
   if width<900
      figwidth = width - 250;
   else
      figwidth = 700;
   end
   if (find(get(0,'child')==2)),
       tolsimu('run     ');
   end
   figpos= get(2,'Position');
```

```
   set(2,...% define plot-window
          'Visible','off',...% define plot-window
          'Units','pixels',...
          'NumberTitle','Off',...
          'Menu','None',...
          'Position',[figposx figposy figheight figwidth],...
          'Color',[0.0 0.0 0.0]);
   bwcon;
   figure(2);
   print -dwinc
   set(2,'Position',figpos,'color','default','Visible','off');
   close(2)
   tolsimu('run     ');
%-------------------------------------------------------%
% Case 9: Call of tolsimu by help-button               %
%                                                      %
% action:                                              %
% --------                                             %
% * call help for tolsimu-function                     %
%                                                      %
%-------------------------------------------------------%
elseif strcmp(action,'chcolm  '),

   if ~isempty(find(get(0,'child')==2)),
     a=get(2,'UserData');
     set(2,'UserData',[a(1) (3-a(2))]);
     bwcon;
   end

%-------------------------------------------------------%
% Case 9: Call of tolsimu by help-button               %
%                                                      %
% action:                                              %
% --------                                             %
% * call help for tolsimu-function                     %
%                                                      %
%-------------------------------------------------------%
elseif strcmp(action,'help    '),
help tolsimu
%-------------------------------------------------------%
% Case 10: Call of tolsimu by the exit-button          %
%                                                      %
% action:                                              %
% --------                                             %
% * close main window                                  %
%                                                      %
%-------------------------------------------------------%
elseif strcmp(action,'exit    '),             % Exit
   if ~(isempty(find(get(0,'child')==2))),
      close(2);
   end
   if mversion==5,
      closereq;
   else
      close(1);
   end
   set(0,'visible','on')
end
```

APPENDIX 4 MATLAB® listing: Transformer Simulation (trafosim.m)

```
%|-------------------------------------------------------|
%|         Philips Semiconductors, SLH                   |
%|-------------------------------------------------------|
%|                                                       |
%| Kernel Transformer simulation for Tolsimu             |
%|                                                       |
%| Version: 1.0                                          |
%|-------------------------------------------------------|
%| File: TRAFOSIM.M                                      |
%| Author(s): W.Tobergte                                 |
%| Date: 26/03/99                                        |
%| Last Change: 29/03/1999                               |
%| --W.Tobergte                                          |
%|-------------------------------------------------------|
if (exist('tol1'))==0,
   tol1=0.10;   % antenna tolerance range
end
if (exist('tol2'))==0,
   tol2=0.10;   % transponder tolerance range
end
if (exist('OSCtol'))==0,
   OSCtol=0;    % carrier-frequency tolerance
end

if (exist('step1'))==0,
   step1 =80;   % antenna frequency steps
end
if (exist('step2'))==0,
   step2 =80;   % transponder frequency steps
end
f0 = 125e3;                     % center frequency for tolerance simulation
w0 = 2*pi*f0*(1+OSCtol/100);    % carrier frequency
fpit = linspace(f0*(1-tol2),f0*(1+tol2),step2);
wpit = 2 * pi * fpit;                                % resonant frequency of PIT
fbasis = linspace(f0*(1-tol1),f0*(1+tol1),step1);
wbasis = 2 * pi * fbasis;                            % resonant frequency of basestation
% Equivalent Circuit
M = k * sqrt(L1 .* L2);
XM = j * w0 * M;
% Antenna of basestation (Variation of resonant frequency at C1)
XL1 = j * w0 * L1;
C1 = ones(size(wpit))' * (1 ./ (wbasis .^ 2 * L1));
XC1 = 1 ./ (j * w0 * C1);
RL1 = w0 * L1 / Q1;
% Transponder (Variation of resonant frequency at C2)
XL2 = j * w0 * L2;
C2 = (1 ./ (wpit .^ 2 * L2))' * ones(size(wbasis));
XC2 = 1 ./ (j * w0 * C2);
RL2 = w0 * L2 / Q2;
% Calculation of equivalent voltage source U0
XLs1 = (L1 - M) .* j .* w0;
XLs2 = (L2 - M) .* j .* w0;
Z11 = RL1 + XLs1 + XC1;
Z22 = RL2 + XLs2 + XC2;
U0 = U1 .* (1 - Z11 ./ (Z11 + par(XM, Z22))) .* XC2 ./ Z22;
Zi = U0 .* (XLs2 + RL2) ./ (U1 .* (1 - Z11 ./ (Z11 + par(XM, XLs2 + RL2))));
```

```
if ~exist('Rclip1')
   Rclip1=0;
end
a = real(Zi)+Rclip1;
b = imag(Zi);
%Rmax = 10e3;
% Calculation of resistor R1 for clipping level Uclip1
n = Uclip1.^2-abs(U0).^2;
p = Uclip1^2*2*a ./ n;
q = (a.^2+b.^2)*Uclip1^2 ./ n;
x = p.^2 / 4 - q;
R1 = -p./2 + sqrt(x);
i = find(R1 < 0);
R1(i) = ones(size(i)) * Rmax;
R1 = min(R1+Rclip1, Rmax); % limitation of R1
Z1 = R1;
%Calculation of first half wafe voltage using RA
U2a=abs(U0.*RA./(Zi+RA));
U2a=min(U2a,Uclip2a); % limiter function
%Calculation of second half wafe voltage using RB
U2b=abs(U0.*RB./(Zi+RB));
U2b=min(U2b,Uclip2b);   % limiter function
Uclip2=(U2a+U2b)/2;
% Calculation of the resistor Z2 for avarage voltage of the loaded state
n = Uclip2.^2-abs(U0).^2;
p = Uclip2.^2.*2.*a ./ n;
q = (a.^2+b.^2).*Uclip2.^2 ./ n;
x = p.^2 / 4 - q;
Z2 = -p./2 + sqrt(x);
i = find(Z2 < 0);
Z2(i) = ones(size(i)) * Rmax;
% Total impedance for transformer equivalant circuit
Zg1 = XLs1 + RL1 + par(XM, XLs2 + RL2 + par(Z1, XC2)) + XC1;
Zg2 = XLs1 + RL1 + par(XM, XLs2 + RL2 + par(Z2, XC2)) + XC1;
% maximum basestation driver current
i1max = U1/min(min(abs(Zg1)))*1e3;
demod;     % calculate demodulation
if ~exist('v')
  mindv = min(min(dv));      % get minimum value of dv
  maxdv = max(max(dv));      % get maximum value of dv
  if exist('demodsense')
    v = [0 demodsense linspace(mindv, maxdv, 3)];  % calculate 5 lines for contour-plot
  else
    v = [0 linspace(mindv, maxdv, 4)];  % calculate 5 lines for contour-plot
  end;
end
if ~(OSCtol==0)
  plttext = sprintf('L1=%4.3fmH, Q1=%2.1f, Q2=%2.1f, k=%3.2f%%, U1=%3.2fV, I1max=%3.0fmA,
f0=%3.1fkHz',...
                    1e3* L1   , Q1      , Q2      , 100*k    , U1       , i1max, w0/2/pi/1000);
else
  plttext = sprintf('L1=%4.3fmH, Q1=%2.1f, Q2=%2.1f, k=%3.2f%%, U1=%3.2fV, I1max=%3.0fmA',...
                    1e3* L1   , Q1      , Q2      , 100*k    , U1       , i1max);
end;
```

APPENDIX 5 MATLAB® listing: Generel parameter list (general.m)

```
%|-------------------------------------------------------|
%|         Philips Semiconductors, SLH                   |
%|-------------------------------------------------------|
%| Genaral simulation parameter                          |
%|-------------------------------------------------------|
%| File: GENERAL.M                                       |
%| Author(s): W.Tobergte                                 |
%| Date: 02/03/97                                        |
%| Last Change: 03/07/1997                               |
%|-------------------------------------------------------|

%-------------------------------------------------------------------------------%
tol1=0.1;    % Antenna tolerance range
tol2=0.1;    % transponder tolerance range
%OSCtol=0;   % carrier-frequency tolerance in %

step1 =100;   % antenna frequency steps
step2 =100;   % transponder frequency steps
%-------------------------------------------------------------------------------%
step1=2*floor(step1/2); %  must be an even number
step2=2*floor(step2/2); %  must be an even number
```

APPENDIX 6 MATLAB® listing: Basestation parameter(bas.m)

```
%|-------------------------------------------------------|
%|         Philips Semiconductors, SLH                   |
%|-------------------------------------------------------|
%| Basestation parameters for tolsimu simulation         |
%|                                                       |
%| The following values _must_ be defined:               |
%|   * k  (coupling-coefficient)                         |
%|   * L1 (Inductivity of basestation)                   |
%|   * U1 (Carrier Supply Voltage)                       |
%|   * Q1 (Quality of basestation)                       |
%|   * RFE (Iron losses)                                 |
%|   * demodsense (Demodulator sensetivit)               |
%|-------------------------------------------------------|
%| File: BAS.M                                           |
%| Author(s): W.Tobergte                                 |
%| Date: 02/03/97                                        |
%| Last Change: 03/07/1997                               |
%|-------------------------------------------------------|
typelist=[];typecount=0;
v=[20 20];
%-------------------------------------------------------------------------------%
if exist('myappl.m')==2
   myappl
end
%-------------------------------------------------------------------------------%
typelist=[typelist;'Example 1      '];
typecount=typecount+1;
if typ == typecount       % ************ Example 1
  k = 0.01;
  U1 = 5*4/pi;
  L1=1000e-6;
  Q1=15;
  RFE = 1e6;
  demodsense =20;
end
%-------------------------------------------------------------------------------%
typelist=[typelist;'Example 2      '];
typecount=typecount+1;
if typ == typecount       % ************ Example 2
  k = 0.006;
  L1 = 1000e-6;
  Q1 = 12.3;
  U1 = 6.37;
  RFE = 1e6;
  demodsense =20;
end
%-------------------------------------------------------------------------------%
typelist=[typelist;'Example 3      '];
typecount=typecount+1;
if typ == typecount       % ************ Example 3
  k = 0.03;
  L1 = 330e-6;
  Q1 = 15;
  U1 = 2.5;
  RFE = 1e9;
  demodsense =20;
end
%-------------------------------------------------------------------------------%
typetext=typelist(typ,:);
```

APPENDIX 7 MATLAB® listing: Transponder parameter list (trans.m)

```
%|-------------------------------------------------------|
%|         Philips Semiconductors, SLH                   |
%|-------------------------------------------------------|
%|                                                       |
%|Transponder parameters for Tolsimu                     |
%|                                                       |
%|The following values _must_ be defined:                |
%| * transminres (min. transponder res.-                 |
%|                frequency in kHz)                      |
%| * transmaxres (max. transponder res.-                 |
%|                frequency in kHz)                      |
%| * L2 (inductivity of transponder)                     |
%| * Q2 (quality of transponder)                         |
%| * Uclip1 (upper clipping level)                       |
%| * Uclip2a,b (lower clipping level)                    |
%| * RA,RB (basic load resistance)                       |
%|                                                       |
%| * Bprog (field-strenght in program mode)              |
%| * Uprog (minimal voltage in program mode)             |
%|-------------------------------------------------------|
%| File: TRANS.M                                         |
%| Author(s): W.Tobergte                                 |
%| Date: 02/03/97                                        |
%| Last Change: 25/07/1997                               |
%|-------------------------------------------------------|
if ~exist('transpondertyp')
  transpondertyp=1;
end
translist=[];transcount=0;
%-------------------------------------------------------------------------------%
if exist('mytrans.m')==2
   mytrans
end
%-------------------------------------------------------------------------------%
translist=[translist;'7930XP        '];
transcount=transcount+1;
if transpondertyp == transcount
% LC parameters
  transminres = 121.25;          % ********* PCF7930XP
  transmaxres = 128.75;
  L2 = 2.33e-3;
  Q2 = 35;
% Load modulation parameters
  Rmax=2e5;   %  input impedance below  the clipping upper level
  Uclip1=9;
  RA=Rmax;
  Uclip2a=3;
  RB=Rmax;
  Uclip2b=3;
% Programming parameters
  Bprog=140e-6;
  Uprog=Uclip1;
  Rprog=min(Rmax,r_prog(Bprog, L2, Q2, Uprog, transminres, 125e3));
end
```

```
%--------------------------------------------------------------------------%
translist=[translist;'79735         '];
transcount=transcount+1;
if transpondertyp == transcount
% LC parameters
  transminres = 121.9;           % ********* PCF79735
  transmaxres = 128.1;
  L2 = 2.33e-3;
  Q2 = 35;
% Load modulation parameters
  Rmax=2e5;   % input impedance below  the clipping upper level
  Uclip1=9;  % Upper clipping level
  RA=Rmax;
  Uclip2a=3;  % Lower clipping level
  RB=Rmax;
  Uclip2b=Uclip2a;
% Programming parameters
  Bprog=100e-6;
  Uprog=Uclip1;
  Rprog=min(Rmax,r_prog(Bprog, L2, Q2, Uprog, transminres, 125e3));
end
%--------------------------------------------------------------------------%
%--------------------------------------------------------------------------%
translist=[translist;'79736 HITAG   '];
transcount=transcount+1;
if (transpondertyp == transcount)
% LC parameters
  transminres = 121;
  transmaxres = 129;
  L2 = 2.33e-3;
  Q2 = 35;
% Load modulation parameters
  Uclip1=6;% Upper clipping level
  RA=4.7e3;
  Uclip2a=Uclip1;
  RB=2.2e3;
  Uclip2b=2;    % lower clipping level
  Rmax=200e3;   %  input impedance below  the clipping upper level
% Programming parameters
  Uprog=4;
  Bprog=35e-6;
  Rprog=min(Rmax,r_prog(Bprog, L2, Q2, Uprog, transminres, 125e3));
end
%--------------------------------------------------------------------------%
```

APPENDIX 8 MATLAB® listing: Demodulation (demod.m)

```
%|-------------------------------------------------------|
%|         Philips Semiconductors, SLH                   |
%|-------------------------------------------------------|
%| Demodulatior models                                   |
%|                                                       |
%| List of available demodulation-methods and their      |
%|    formulas                                           |
%|                                                       |
%| For the calculation of the demodulation all data      |
%|    calculated in tolsimu.m are available,             |
%|    especially Zg1 and Zg2 !!!                         |
%|                                                       |
%| The output of dv must be a matrix!                    |
%|                                                       |
%| The value of demodsense defines the minimum input     |
%|     value of dv (optional)                            |
%|                                                       |
%| The values of the vector v define the contour-lines in |
%|     the contour-plot (optional)                       |
%|-------------------------------------------------------|
%| File: DEMOD.M                                         |
%| Author(s): W.Tobergte                                 |
%| Date: 02/03/97                                        |
%| Last Change: 03/07/1997                               |
%|-------------------------------------------------------|
if ~exist('demodtyp')
  demodtyp=0;
end
demodlist=[];demodcount=0;
%-------------------------------------------------------------------------------%
if exist('mydemod.m')==2
   mydemod
end
%-------------------------------------------------------------------------------%
demodlist = [demodlist;'Envelope of Uc [mV]         '];
demodcount=demodcount+1;
if demodtyp == demodcount       % envelope Uc[mV]
    dv = 1e3 * ( abs(U1 .* XC1 ./ Zg2) - abs(U1 .* XC1 ./ Zg1));
%    demodsense = 40;
end
%-------------------------------------------------------------------------------%
demodlist = [demodlist;'AST Demodulation (ABIC)     '];
demodcount=demodcount+1;
if demodtyp == demodcount       %  AST demodulator
    demodsense = 2;  % 2 mV
%   calculation of the phase shift of the antenna input impedance
    Phi1=angle(XC1+XL1+RL1);
%   64 quantisation steps
    quant=8/0.125;  % 8Mhz/125kHz =64 steps
%   Caculation of the adaptive phase shift Phi_Q
%   DeltaN : quantisation method coefficient
%   DeltaN = 0 : round down
%   DeltaN = 1 : standard round
%   DeltaN = 2 : round up
%
    DeltaN=1;
    Phi_S=pi/2  + (2*floor(Phi1/(2*pi)*quant)+DeltaN)/quant*2*pi;
%   voltage deviations of Uc1*exp(j*Phi_S)
    dv = 1e3 * real( U1 .* XC1 .*exp(j.*Phi_S).*(1./ Zg2-1./ Zg1) );
%   Voltage divider
    Urec_max=8;
```

```
    k_div=Urec_max/Q1/U1;
    dv=k_div*dv;
%   contour lines at v= [y1 y2 ....]
%   special single contour lines  v= [y1 y1];
    v=[0 demodsense];
end
%-------------------------------------------------------------------------%
demodlist = [demodlist;'Worst case cont. AST        '];
demodcount=demodcount+1;
if demodtyp == demodcount       %  AST demodulator
    demodsense = 2;  % 2 mV
%   calculation of the phase shift of the antenna input impedance
    Phi1=angle(XC1+XL1+RL1);
%   64 quantisation steps
    quant=8/0.125;  % 8Mhz/125kHz =64 steps
%   Caculation of the adaptive phase shift Phi_Q
%   DeltaN : quantisation method coefficient
%   DeltaN = 0 : round down
%   DeltaN = 1 : standard round
%   DeltaN = 2 : round up
%
    DeltaN=1;
    Phi_S1=pi/2  + 2*Phi1-pi/32;
    Phi_S2=pi/2  + 2*Phi1+pi/32;
    Phi_S=[ Phi_S1(1:step2/2,:) ; Phi_S2(step2/2+1:step2,:) ];
%   voltage deviations of Uc1*exp(j*Phi_S)
    dv = 1e3 * real( U1 .* XC1 .*exp(j.*Phi_S).*(1./ Zg2-1./ Zg1) );
%   Voltage divider
    Urec_max=8;
    k_div=Urec_max/Q1/U1;
    dv=k_div*dv;
%   contour lines at v= [y1 y2 ....]
%   special single contour lines  v= [y1 y1];
    v=[0 demodsense];
end
%-------------------------------------------------------------------------%
demodlist = [demodlist;'Phase of Uc                 '];
demodcount=demodcount+1;
if demodtyp == demodcount       % phase of Uc
    dv = 1e3 * abs(atan(imag(U1 .* XC1 ./ Zg2)./real(U1 .* XC1 ./ Zg2))...
         - atan(imag(U1 .* XC1 ./ Zg1)./real(U1 .* XC1 ./ Zg1)));
    demodsense = 40;
end
%-------------------------------------------------------------------------%
demodlist = [demodlist;'Real Part of Quadratur      '];
demodcount=demodcount+1;
if demodtyp == demodcount       % Real
    dv = 1e3 * abs(( real(U1 .* XC1 ./ Zg2) - real(U1 .* XC1 ./ Zg1)));
end
%-------------------------------------------------------------------------%
demodlist = [demodlist;'Imaginary Part of Quadratur '];
demodcount=demodcount+1;
if demodtyp == demodcount       % Imag
    dv = 1e3 * abs(( imag(U1 .* XC1 ./ Zg2) - imag(U1 .* XC1 ./ Zg1)));
end
```

```
%-------------------------------------------------------------------------%
demodlist = [demodlist;'Quadratur, Max(Re, Im)      '];
demodcount=demodcount+1;
if demodtyp == demodcount       % Quadrature
    RealPart = abs(( real(U1 .* XC1 ./ Zg2) - real(U1 .* XC1 ./ Zg1)));
    ImagPart = abs(( imag(U1 .* XC1 ./ Zg2) - imag(U1 .* XC1 ./ Zg1)));
    dv = 1e3 * max(RealPart, ImagPart);
    demodsense = 40;
end
%-------------------------------------------------------------------------%
demodlist = [demodlist;'Complex Difference Demod.   '];
demodcount=demodcount+1;
if demodtyp == demodcount       % complex difference
    dv = 1e3 * abs((U1.*XC1./Zg2) - (U1.*XC1./Zg1));
end
%-------------------------------------------------------------------------%
demodlist = [demodlist;'Voltage at Rprog            '];
demodcount=demodcount+1;
if demodtyp == demodcount       % Voltage at Rprog
  dv=abs(U0.*Rprog./(Zi+Rprog));
    v=[ Uprog Uprog];
    demodsense=Uprog;
end
```

APPENDIX 9 MATLAB® listing: Plot functions (plotfun.m)

```
%|-------------------------------------------------------|
%|         Philips Semiconductors, SLH                   |
%|-------------------------------------------------------|
%| Function for displaying the simulation results        |
%|-------------------------------------------------------|
%| File: TRANS.M                                         |
%| Author(s): W.Tobergte                                 |
%| Date: 02/03/97                                        |
%| Last Change: 03/07/1997                               |
%|-------------------------------------------------------|
% axis scaling
if exist('plottyp')
   set(1,'units','norm');
   scpos=get(1,'pos');
   scpos(1)=scpos(1)+scpos(3)+0.01;
   scpos(3)=(1-scpos(1)-0.01);
   scpos(2)=(scpos(2)+scpos(4)-1*scpos(3));
   if scpos(2)<0.01;
       scpos(2)=0.01;
   end
   scpos(4)=scpos(3);
   skala1 = (fbasis - f0) ./ f0 * 100;
   skala2 = (fpit - f0) ./ f0 * 100;
   x_pl = tol1 * (-100);
   y_pl = tol2 * (-120);
   xt = skala1(1);
   yt = skala2(length(skala2))*1.08;
%  selection of Plot window
   if (isempty(find(get(0,'child')==pic))),
      figure(pic);
      set(pic...              % define plot-window
         ,'Units','norm'...%
         ,'papertyp','A4'...
         ,'Visible','off'...
         ,'NumberTitle','Off'...
         );

      set(pic,'Position',scpos);     %Hier scpos
      set(gca,'Color',0*[1 1 1],'box','on');
      set(gcf,'paperorient','landscape');
      set(gcf,'papertype','A4');
      colortyp=1;
      set(pic,'UserData',[plottyp colortyp]);
      b=(0.7*(0:63)/63);
      a=1-b'*[0 0 1];
      %colormap(a);
      colormap('default');
      set(pic,'UserData',[99999 1]);
%
   else
       figure(pic);
   end
else
  plottyp=0;
end
```

```
% Begin of Plot List %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
zoom on
plotlist=[];plotcount=0;
%----------------------------------------------------------------------------%
if exist('myplot.m')==2
   myplot
end
%----------------------------------------------------------------------------%
%----------------------------------------------------------------------------%
plotlist=[plotlist;'Contour       '];
plotcount=plotcount+1;
if plottyp == plotcount
  figure(pic);
  clf;
  set(pic,...  % define plot-window
   'Name','Pit tolerance simulation - Contour Plot','visible','on');
  [Mod,h] = contour(skala1, skala2, dv,v);
  clabel(Mod);
  set(gca,'layer','top','box','on');
  h=text(0, +(tol2*90), plttext, 'HorizontalAlignment','center');
  set(h,'FontSize', 8)
  if transminres & transmaxres
     x = (transminres - 125) / 125 * 100 ;% calculate min. tol.-limit
     y = (transmaxres - 125) / 125 * 100 ;% calculate max. tol.-limit
     hold on
     plot([-(100*tol1) 100*tol1],[x x],'--',...
        [-(100*tol1) 100*tol1],[y y],'--');% plot line for min.-limit
     t=text(-tol1*99, x-tol2,sprintf('%3.1f%%',x));
     set(t,'horizontal','left');
     set(t,'vertical','top');
     t=text(-tol1*99, y+tol2,sprintf('%3.1f%%',y));
     set(t,'horizontal','left');
     set(t,'vertical','bottom');
     hold off
  end
  % antenna tolerance limits
  if exist('demodsense')
    % Calculation of SAFE OPERATING AREA:
    SOARmin=-tol1*100;
    SOARmax=tol1*100;
    % get all rows of dv, where the transponder-frequency is between the possible tolerances:
    for row = min(find(fpit>=transminres*1000)) : max(find(fpit<=transmaxres*1000))
      soar=skala1( max( find( dv(row,1:step1/2) < demodsense ) ) +1 );
      if soar>SOARmin
        SOARmin=soar;
      end
      soar=skala1( step1/2 + min( find( dv(row,step1/2+1:step1) < demodsense ) ) -1 );
      if soar<SOARmax
        SOARmax=soar;
      end
    end
    basisminres=125*(1+SOARmin/100);
    basismaxres=125*(1+SOARmax/100);
    % Calculation of tolerance limits:
    x = (basisminres - 125) / 125 * 100;% calculate min. tol.-limit
    y = (basismaxres - 125) / 125 * 100;% calculate max. tol.-limit
    hold on
    plot([x x], [-(100*tol2) 100*tol2],'--');% plot line for min.-limit
    plot([y y], [-(100*tol2) 100*tol2],'--');% plot line for max.-limit
    t=text(x-tol1,-tol2*99,sprintf('%3.1f%%',x));
    set(t,'horizontal','right');
    set(t,'vertical','bottom');
    t=text(y+tol1,-tol2*99,sprintf('%3.1f%%',y));
```

```
    set(t,'horizontal','left');
    set(t,'vertical','bottom');
    hold off
  end
  demodtext=demodlist(demodtyp,:);% get demodtyp for plot-label
  title([demodtext 'Appl.: '  typetext 'Transp.: ' translist(transpondertyp,:)]);
  grid on;
  xlabel('basestation freq. detuning [%]');
  ylabel('transponder freq. detuning [%]');
end
%-------------------------------------------------------------------------%
plotlist=[plotlist;'Add Contour   '];
plotcount=plotcount+1;
if plottyp == plotcount

  a=get(gcf,'UserData');
  if  (a(1)>(plottyp)),

      plottyp=plottyp-1;  % call "contour" instead of "add contour" if figure 2 is closed
      plotfun;
  else
      hold on
      figure(pic);
      labeltype=get(gca,'UserData');
      if isempty(labeltype)
      labeltype = '--';
  elseif labeltype == '--'
      labeltype = '-.';
  else
      labeltype = ':';
      end;
      set(gca,'UserData',labeltype);
      Mod = contour(skala1, skala2, dv, 3, labeltype);
      clabel(Mod);
%      set(gca,'color',[1 1 1]);
      hold off
   end
end
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
plotlist=[plotlist;'Shaded Contour'];
plotcount=plotcount+1;
if plottyp == plotcount
  figure(pic);
  clf;
  colormap('gray');
  set(pic,...% define plot-window
   'Name','Pit tolerance simulation - Shaded Contour Plot');
  clf;
  hold on
  if exist('demodsense')
    zline=demodsense;
  else
    zline=0;
  end;

  if findstr(version,'5.')==1,
     shift_line=0;
  else
     shift_line=1;
  end
  demodsense=min(min(dv));
  surf(skala1, skala2, dv -1e10*shift_line);
```

```
  shading flat;
  view(0,90);
  a=get(gcf,'currentaxes');
  xticks=get(a,'xtick');
  yticks=get(a,'ytick');
  [xgrid ygrid]=meshgrid(xticks,yticks);
  t=mesh(xgrid,ygrid,zeros(length(yticks),...
         length(xticks)) + zline*0.999 -1e10*shift_line);
  set(t,'edgecolor',0.5*[1 1 1],'facecolor',0.9*[1 1 1]);

  [Mod,h] = contour(skala1, skala2,dv, 5);
  clabel(Mod);
  set(h,'erasemode','xor','LineWidth',1.5);
  demodtext=demodlist(demodtyp,:);% get demodtyp for plot-label

  title([demodtext 'Appl.: '  typetext 'Transp.: ' translist(transpondertyp,:)]);
  grid on;
  xlabel('basestation freq. detuning [%]');
  ylabel('transponder freq. detuning [%]');
  set(gca,'color',[1 1 1]);
  hold off
end
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
plotlist=[plotlist;'3D Mesh Grid  '];
plotcount=plotcount+1;
if plottyp == plotcount
  figure(pic);
  clf;
  colormap('default');
  set(pic,...% define plot-window
   'Name','Pit tolerance simulation - 3D Mesh Grid');
  surf(skala1, skala2, dv+10);
  view(viewmtx(-30,30,30));
  demodtext=demodlist(demodtyp,:);% get demodtyp for plot-label
  title([demodtext 'Appl.: '  typetext 'Transp.: ' translist(transpondertyp,:)]);
  grid on;
  xlabel('basestation freq. detuning [%]');
  ylabel('transponder freq. detuning [%]');
end
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
%-------------------------------------------------------------------------%
```

APPENDIX 10 MATLAB® listing: Equivalent resistance in Program Mode (r_prog.m)

```
%|-------------------------------------------------------|
%|         Philips Semiconductors, SLH                   |
%|-------------------------------------------------------|
%|                                                       |
%| Function for the calculation of the equivalent        |
%| IC resistance of the transponder                      |
%|                                                       |
%|-------------------------------------------------------|
%| File: R_PROG.M                                        |
%| Author(s): W.Tobergte                                 |
%| Date: 02/03/97                                        |
%| Last Change: 03/07/1997                               |
%|-------------------------------------------------------|
function [Rprog] = r_prog(Bprog, L2, QT, Uclip1, transmaxres, f0)
tol = transmaxres/125 - 1;              % frequency tolerance
U2=Uclip1;                          % upper clipping level
w0=2*pi*f0; %
R2 = w0*L2/QT; %
gamma = 20e3;                           % proportional factor for Uq in dependency of B
C2 = 1/(((1+tol)*w0)^2)/L2;         % resonant frequency detuning of tol[%]
XC2=1/(j*w0*C2);
XL2=j*w0*L2;
Uq=gamma.*Bprog/2;                      % calculate the induced voltage
U0=abs(Uq./(1+j*w0*R2*C2-w0^2*L2*C2));   % calculate the equivalent open circuit voltage
if (U0<Uclip1)
   fprintf('***** Error on Rprog Calculation\n\a')
   Rprog=0;
else
   Zi=((XC2*(XL2+R2))/(XC2+XL2+R2));     % calculate the equivalent internal resistance
   Rprog = -(real(Zi)*U2^2)./(U2^2-U0.^2)...
        + sqrt(((real(Zi)*U2^2)./(U2^2-U0.^2)).^2-(U2^2*abs(Zi)^2)./(U2^2-U0.^2));
end
```

APPENDIX 11 MATLAB® listing: Parallel impedances function (par.m)

```
function Z = par(Z1, Z2, Z3);
if nargin >= 2
  Z = Z1 .* Z2 ./ (Z1 + Z2);
end
if nargin >= 3
  Z = Z .* Z3 ./ (Z + Z3);
end
```

APPENDIX 12 MATLAB® listing: Black&white style conversion (bwcon.m)

```
%|-------------------------------------------------------|
%|         Philips Semiconductors, SLH                   |
%|-------------------------------------------------------|
%|                                                       |
%| Function for converts style of Matlab objects to a    |
%|          proper document form:                        |
%|          white background                             |
%|          black text, lines, labels, etc               |
%|-------------------------------------------------------|
%| File:  BWCON.M                                        |
%| Author(s): R.Lueneberg W.Tobergte                     |
%| Date: 02/03/94                                        |
%| Last Change: 29/03/1999                               |
%|-------------------------------------------------------|
% Figure 2 is valid ??

if isempty(find(get(0,'child')==2)),
   return;
else
   figure(2);
   pl_data=get(2,'UserData');
end

%


%Black2white converion for Matlab version 4.x
if mversion==4
    copyfigure=2;
    set( copyfigure,...
         'InvertHardcopy','off',...
         'Color',[1 1 1],'visible','off');
    ax=get(copyfigure,'Children');%get axes
    if ax ~= []
      for j=1:length(ax),
       if strcmp(get(ax(j),'Type') , 'axes')
         set(ax(j),...
            'Box','on',...
            'FontSize',[12],...
            'XColor',[0 0 0],...
            'YColor',[0 0 0],...
            'ZColor',[0 0 0]);
         lntx=get(ax(j),'Children');%get lines, text & surface
         xlbl=get(ax(j),'XLabel');%get xlabel
         ylbl=get(ax(j),'YLabel');%get ylabel
         zlbl=get(ax(j),'ZLabel');%get zlabel
         ttl=get(ax(j),'Title');%get title
         if lntx ~= []
           for k=1:length(lntx),
                if (strcmp(get(lntx(k),'Type') , 'surface')==0),
                     set(lntx(k),'Color',[0 0 0]);
                else
                   if(strcmp(get(lntx(k),'FaceColor') , 'interp')==0),
                        set(lntx(k),'FaceColor',0.9*[1 1 1]);
                   end
                end;
           end;
         end;
         if ylbl ~= []
         set(ylbl,'Color',[0 0 0]);
         end;
```

```
        if zlbl ~= []
           set(zlbl,'Color',[0 0 0]);
        end;
        if ttl ~= []
           set(ttl,'Color',[0 0 0]);
        end;
      end;
    end;
  end;
  set( copyfigure,'visible','off');
end
%

%Blackwhit converion for Matlab version 5

if mversion==5,
    if pl_data(1)<3,        % only for countour plot
         if  pl_data(2)==2,   % Black White Mode
            colormap('gray');
            linecolor=[0 0 0];
            fcolor=[0 0 0];
            ach=get(gca,'child');
            scrsize=get(0,'screensize');
            lw=scrsize(3)/800;  %set Line width
            set(ach(find(strcmp(get(ach,'type'),...
               'patch'))),'edgecolor',fcolor,...
               'Linewidth',lw);
            set(ach(find(strcmp(get(ach,'type'),'line'))),...
               'color',fcolor,...
               'Linewidth',lw);
          elseif pl_data(2)==1, % Standart Color Mode
            ach=get(gca,'child');
            fcolor=[0 0 1];
            scrsize=get(0,'screensize');
            lw=scrsize(3)/800;  %set Line width
            achpatch=ach(find(strcmp(get(ach,'type'),'patch')));
            if ~isempty(achpatch),
              colormap('default');
              fcolor=colormap;
              a=get(achpatch,'userdata');
              if length(a)>1,
                eval('b=cat(1,5,a{:});');
              else
                b=[a a*1.00001];
                achpatch=[achpatch achpatch];
              end
              bmax=max(b);

              for i=1:length(achpatch);
                 a=(get(achpatch(i),'Userdata')-min(b))/(max(b)-min(b));
                 set(achpatch(i),...
                    'edgecolor',fcolor(floor(50*a+10),:),...
                    'linewidth',lw...
                    );
              end
            end
            achline=ach(find(strcmp(get(ach,'type'),'line')));
            k=1;
            if ~isempty(achline),
              colormap('default');
              fcolor=colormap;
              if length(achline)==1,
                   achline=[achline achline];
```

```
            end
            for i=1:length(achline);
               set(achline(i),...
                  'color',fcolor(k,:)*0.7,...
                  'linewidth',lw...
                  );
               k=k+10;
               if k>64, k=k-64;end;
            end
         end
      end
   elseif   pl_data(1)==4  % 3D Plot
      if  pl_data(2)==2,   % Standart Color Mode
         colormap('gray');
      elseif pl_data(2)==1, % Black WhiteMode -> Standart
         colormap('default');
      end
   end
end
```