



Narrow band applications

Version 1.0

| | | |
|----------|---------|--------------------------|
| Document | Type | |
| Document | Status | |
| Document | Version | Version 1.0 |
| Product | | Narrow band applications |

Table of Contents

| | |
|--|-----------|
| 1. Narrow band applications | 4 |
| 1.1. Overview | 4 |
| File structure | 4 |
| 1.2. Introduction | 5 |
| What is the advantage of narrow band? | 5 |
| Drawbacks of narrow band applications..... | 5 |
| 1.3. What is frequency acquisition good for and how does it work?..... | 6 |
| Calculating the frequency offset | 7 |
| 1.4. Step by step instruction..... | 8 |
| Prepare register values | 8 |
| Modify and run the sample code | 11 |
| 1.5. Test the performance of your system..... | 13 |
| Performance testing and tuning | 14 |
| limitations..... | 14 |
| 2. Appendix..... | 15 |
| 2.1. References | 15 |
| AX5042 downloads..... | 15 |
| 3. Contact Information..... | 16 |


















1. Narrow band applications

1.1. Overview

This document provides useful information about the usage of the AX5042 in narrow band applications. Read chapter 0 and 1.3 to get basic information about narrow band applications and to learn what it's good for. Chapter 1.3 also provides formulas and theoretical background information. In chapter 1.4 you will find a step by step instruction how to calculate the required register settings and how to modify the source code. Finally a test setup is suggested in chapter 1.5 which can be used to measure the performance of your system.

File structure

This section describes the structure of the files associated with this application note. Please note that only files are listed that are relevant for this documentation.

| | | |
|---|---------------------------------|---|
|  | documentation | |
|  | documentation.pdf | this document |
|  | Calculate Register Settings.xls | Excel sheet to calculate register settings |
|  | firmware | |
|  | main_rx.hex | hex file for 868.3MHz, 4.8kbps receiver |
|  | main_tx.hex | hex file for 868.3MHz, 4.8kbps transmitter |
|  | source | project root directory |
|  | RX_5042 | project folder for receiver |
|  | main_rx.wsp | project workspace for SilconLabs IDE |
|  | main_rx.c | main file |
|  | TX_5042 | project folder for transmitter |
|  | main_tx.wsp | project workspace for SilconLabs IDE |
|  | main_tx.c | main file |
|  | 5042_4_8k_fsk_868.3MHz.h | register settings for FSK 4.8kbps, 868.3MHz |
|  | 5042_4_8k_fsk_916.5MHz.h | register settings for FSK 4.8kbps, 916.5MHz |
|  | 5042_4_8k_fsk_433.0MHz.h | register settings for FSK 4.8kbps, 433.0MHz |
|  | .. | save your own register settings here |

1.2. Introduction

In narrow band applications, frequency offsets between transmitter and receiver caused by crystal drifts and aging have larger impact compared to wide band applications and can lead to bad performance. There are two possibilities to build reliable narrow band communication systems:

- using high precision TCXO crystals to avoid frequency offsets (expensive)
- using frequency acquisition to compensate frequency offsets

This document describes the purpose and functionality of frequency acquisition and provides all required information to build narrow band applications with the AX5042 narrow band transceiver.

What is the advantage of narrow band?

There is one big reason for the usage of narrow band signals: Higher sensitivity. Reducing the signal bit rate results in a narrower spectrum and thus the receivers filter bandwidth can be reduced while the signal is still able to pass through. The reduction of the receivers filter bandwidth results in a lower noise level and makes data reception possible even if the signal is very weak.

Drawbacks of narrow band applications

The reduction of the bit rate has several draw backs however:

Long response time

With low bitrates, it takes longer to transmit the same amount of data than with higher bitrates. For some applications which require fast data transmission such as Walkie Talkies this is not applicable.

Higher current consumption

Because the time required to transmit data is longer, both transmitter and receiver must be switched on longer which is a problem for battery powered devices.

Crystal drift

As already mentioned above, crystal drifts start to play a significant role in narrow band applications and lead to bad performance.

1.3. What is frequency acquisition good for and how does it work?

The frequency offset between a transmitter and receiver depends on the carrier frequency and the crystal accuracy. Use Formula 1 to calculate the maximum frequency offset (worse case) in Hz between a transmitter and receiver.

$$\Delta f_{\max} = 2 \cdot \frac{c}{10^6} \cdot f_{\text{carrier}}$$

Formula 1 maximum frequency offset calculation

f_{carrier} is the operating frequency in Hz, c is the crystal accuracy in ppm. If f_{carrier} is set to 868MHz and c is set to 10ppm, the resulting frequency offset is 17.36kHz. This frequency offset must be compensated by the receiver, otherwise communication is not possible.

The AX5042 automatically compensates frequency offsets up to $\pm 1/2$ of the bit rate (FSK only). Use Formula 2 to calculate the crystal accuracy that is required when the bit rate and carrier frequency are given. If your crystal used does not meet the required accuracy, then you probably need to implement frequency acquisition in your project.

$$c = \frac{\Delta f_{\max} \cdot 10^6}{2 \cdot f_{\text{carrier}}} = \frac{\text{BITRATE} \cdot 10^6}{2 \cdot f_{\text{carrier}}}$$

Formula 2 calculating the required crystal accuracy in ppm (for FSK only)

With a bit rate of 4.8kbps and a carrier frequency of 868MHz, a 2.7ppm crystal is required to guarantee reliable communication, but low cost crystals are in the range of 20ppm which is not good enough and the automatic frequency compensation of the AX5042 will therefore not work anymore.

This is the point, where frequency acquisition comes in. With frequency acquisition, the receiver starts with a bandwidth 8x higher than the nominal bandwidth. This results in a higher bit rate and the automatic frequency tracking circuitry of the AX5042 can now compensate up to ± 46 kHz. After this 'coarse' frequency acquisition, the bandwidth is reduced to approximately 3x the nominal bandwidth and the 'fine' frequency acquisition is started which can compensate up to ± 17 kHz. This is all done in software and now the receiver is ready to receive the actual data with the nominal settings. This process is illustrated in Figure 1.

Because the frequency tracking circuitry of the AX5042 still works with signal levels 10dBm below the level where data reception is possible, the frequency acquisition described above does not lead to loss of sensitivity even if the Rx filter bandwidth was increased.

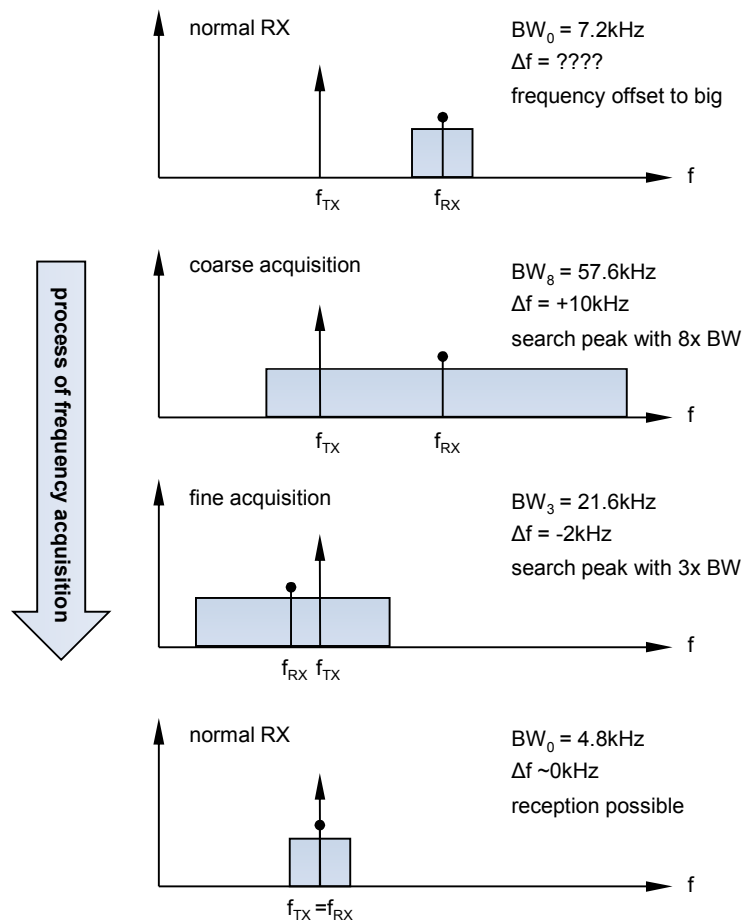


Figure 1 Process of frequency acquisition

Calculating the frequency offset

When reading the TRKFREQ register the frequency offset in Hz can be calculated with Formula 3

$$\Delta f = \left(\frac{BITRATE}{2^{16}} \right) TRKFREQ$$

Formula 3 frequency offset calculation in Hz

The value that must be added to the FREQ register to compensate the frequency offset is calculated by Formula 4. The factor in brackets is constant for every bit rate and can be pre-calculated. The multiplication can then be replaced by shifting the TRKFREQ value accordingly.

$$\Delta FREQ = \left(\frac{2^8 \text{BITRATE}}{f_{XTAL}} \right) TRKFREQ$$

Formula 4 FREQ correction factor calculation

Example:

BITRATE = 4.8kbps, fxtal = 16MHz, the resulting factor is then 0.0768. The calculation in Formula 4 can be replaced with the following code that uses bit shifts only:

```
1 int16_t offs, trkfreq;
2 trkfreq = spi_read16(AX5042_REG_TRKFREQHI); // get TRKFREQ offset
3 offs = trkfreq >>4; // replace multiplication
4 offs += trkfreq >>6; // by factor 0.0768 with
5 offs -= trkfreq >>9; // simple bit shifts
```

Code 1 realize multiplication by 0.0762 with simple bit shifts

The bit shifts in Code 1 are equal to the multiplication by the polynome $2^{-4} + 2^{-6} - 2^{-9}$ which results in a factor of ~ 0.0762 and is accurate enough. The resulting value stored in the variable offs can now be added to the FREQ3..0 registers.

1.4. Step by step instruction

Note: This works only with FSK modulation, since frequency acquisition requires a steady carrier (ASK is only 50% on). First calculate the register values used for frequency acquisition offline, this is described in chapter "Prepare register values". After this, read the chapter "Modify and run the sample code" to learn how to replace the register values in the sample code and run the project. The flow of the sample code is also described in this chapter.

Prepare register values

1: DEFINE NOMINAL RF PARAMETERS

In the first step, all parameters for the normal operation are defined and the register settings are calculated with the AXEVK software[1] by using the "code snippets" function. For this step by step example, the follow parameters are choosen:

nominal bit rate $\text{BITRATE}_0 = 4.8\text{kbps}$
 Modulation: FSK, 868MHz
 Modulation factor h: 0.5

The resulting parameters calculated by the AXEVK are listed below:

nominal bandwidth $\text{BW}_0 = 7.2\text{kHz}$
 CIC decimation factor $\text{CICDEC}_0 = 0x015b$

2: CALCULATE REGISTER SETTINGS FOR COARSE FREQUENCY ESTIMATE

In the second step, the register settings for the coarse frequency estimation are calculated by hand. At first calculate the receiver bandwidth BW_8 for coarse estimation:

$$BW_8 = 8 \cdot BW_0$$

Formula 5 calculation of the receiver bandwidth for coarse frequency estimate

Then write the following registers:

set the register MODULATION₈ to 0x08

set encoding register ENCODING₈ to 0x00

set framing register FRAMING₈ to 0x00

set the data rate register DATARATE₈ to 0x1000

set the receiver register CICDEC₈ value for coarse estimate:

$$CICDEC_8 = \left\lfloor \frac{1.5 \cdot f_{XTAL}}{8 \cdot 1.2 \cdot BW_8} \right\rfloor$$

Formula 6 calculation of CICDEC register for coarse frequency estimate

Re-calculate the receivers bit rate BITRATE₈. Note that the DATARATE₈ is set to 0x1000, which results in a datarate being tied to the filter bandwidth, and having no relationship to the actual transmission datarate. BITRATE₈ is not a register and is only used for further calculations.

$$BITRATE_8 = \frac{2^{10} \cdot f_{XTAL}}{DATARATE_8 \cdot CICDEC_8 \cdot FSKMUL} = \frac{2^{10} \cdot f_{XTAL}}{0x1000 \cdot CICDEC_8 \cdot 1}$$

Formula 7 calculation of the BITRATE value

set the value for AGCATTACK₈

$$AGCATTACK_8 = 27 + \left\lfloor \log_2 \left(\frac{BITRATE_8}{f_{XTAL}} \right) \right\rfloor$$

Formula 8 calculation of the AGCATTACK register value

set the value for AGCDECAY₈

$$AGCDECAY_8 = 27 + \left\lfloor \log_2 \left(\frac{BITRATE_8}{10 \cdot f_{XTAL}} \right) \right\rfloor$$

Formula 9 calculation of the AGCDECAY register value

set the timing recovery register TMGGAIN₈ to 0x0000

set the phase recovery register PHASEGAIN₈ to 0x03

set the register FREQGAIN₈ to 0x6

set the register FREQGAIN2₈ to 0xa

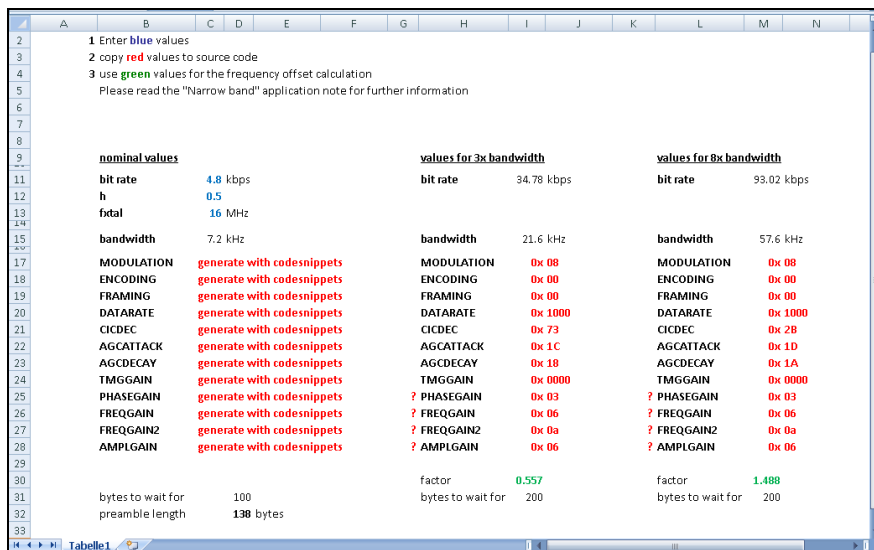
set the register AMPLGAIN₈ to 0x6

3: CALCULATE REGISTER SETTINGS FOR FINE FREQUENCY ESTIMATE

In the third step the register values for the fine frequency estimation are calculated. Use the same procedure as described for the coarse frequency estimation, but this time use $BW_3 = 3 \cdot BW_0$

USING EXCEL SHEET TO OBTAIN REGISTER SETTINGS

Instead of calculating all the register values by hand, the usage of the Excel sheet is recommended. Figure 2 shows a screen shot of the Excel sheet.



| nominal values | | values for 3x bandwidth | | values for 8x bandwidth | |
|-----------------------------------|----------------------------|-------------------------|------------|-------------------------|------------|
| bit rate | 4.8 kbps | bit rate | 34.78 kbps | bit rate | 93.02 kbps |
| h | 0.5 | | | | |
| fxtal | 16 MHz | | | | |
| bandwidth | 7.2 kHz | bandwidth | 21.6 kHz | bandwidth | 57.6 kHz |
| MODULATION | generate with codesnippets | MODULATION | 0x 08 | MODULATION | 0x 08 |
| ENCODING | generate with codesnippets | ENCODING | 0x 00 | ENCODING | 0x 00 |
| FRAMING | generate with codesnippets | FRAMING | 0x 00 | FRAMING | 0x 00 |
| DATARATE | generate with codesnippets | DATARATE | 0x 1000 | DATARATE | 0x 1000 |
| CICDEC | generate with codesnippets | CICDEC | 0x 73 | CICDEC | 0x 28 |
| AGCATTACK | generate with codesnippets | AGCATTACK | 0x 1C | AGCATTACK | 0x 1D |
| AGCDECAY | generate with codesnippets | AGCDECAY | 0x 18 | AGCDECAY | 0x 1A |
| TMGGAIN | generate with codesnippets | TMGGAIN | 0x 0000 | TMGGAIN | 0x 0000 |
| PHASEGAIN | generate with codesnippets | ? PHASEGAIN | 0x 03 | ? PHASEGAIN | 0x 03 |
| FREQGAIN | generate with codesnippets | ? FREQGAIN | 0x 06 | ? FREQGAIN | 0x 06 |
| FREQGAIN2 | generate with codesnippets | ? FREQGAIN2 | 0x 0a | ? FREQGAIN2 | 0x 0a |
| AMPLGAIN | generate with codesnippets | ? AMPLGAIN | 0x 06 | ? AMPLGAIN | 0x 06 |
| bytes to wait for preamble length | 100 | factor | 0.557 | factor | 1.488 |
| | 138 bytes | bytes to wait for | 200 | bytes to wait for | 200 |

Figure 2 Screenshot of Excel sheet

At first, replace the parameters marked in blue with the desired settings.

The red values are then calculated by the Excel sheet and can be copied to the source code.

The green values are required to calculate the frequency offset and the correction value that must be added to the `FREQ3..FREQ0` registers. For coarse frequency acquisition, the factor on the right must be used, for fine frequency acquisition use the factor on the left. Have a look at Code 1 to see how to use these factors in the source code calculation.

The preamble length calculated at the bottom is the minimum preamble that must be sent by the transmitter. It is recommended to start with a preamble much longer at first, and then slowly decrease the preamble length until the optimum is reached.

Modify and run the sample code

After all register values are calculated, the values can now be copied to the sample code. This section describes how to modify the sample code to make the transmitter and receiver running.

RECEIVER

1. Save the header file generated in the section "1: Define nominal RF parameters" in the project root directory
2. Open the main_rx.c file and perform the following changes:
To make sure the header file previously saved in the root directory is included in the project find the function `init_ax5042()` and change the line where the header file is included accordingly
3. Change all register values in the function `SetCoarseAcquisition()` to the values calculated in the section "2: calculate register settings for coarse frequency estimate"
4. Change all register values in the function `SetFineAcquisition()` to the values calculated in the section "3: calculate register settings for fine frequency estimate"
5. compile and run the modified sample code

TRANSMITTER

1. Make sure the header file generated with the code snippets function is stored in the project root directory
2. Open the main_tx.c file and find the function `init_ax5042()`. Change the line containing the `#include` statement so that your header file is included here
3. compile and run the project

The basic program flow of transmitter and receiver are illustrated in Figure 3 and Figure 4. For detailed information and implementation please refer to the sample source code.

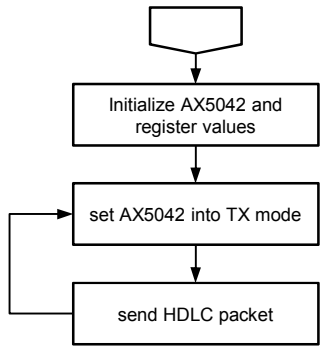


Figure 3 Transmit flow

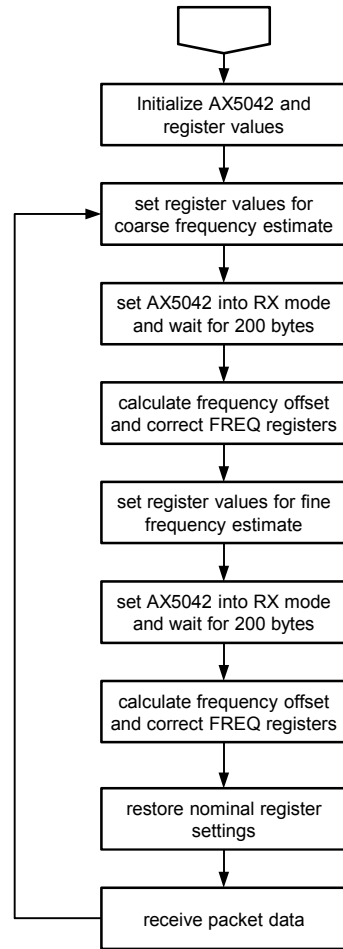


Figure 4 Receive flow

1.5. Test the performance of your system

Now both receiver and transmitter should run and the receiver indicates every received packet by toggling the green LED and every lost packet by toggling the red LED. To measure the sensitivity of the receiver some lab equipment is required. Read this chapter for a suggested sensitivity measurement setup, the components used are illustrated in Figure 1. The sensitivity of the receiver is tested at 10% PER.

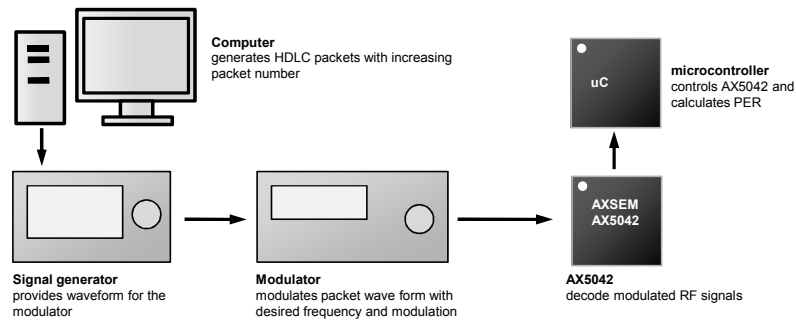


Figure 1 Test setup

Computer

The computer is used to generate the HDLC packet bits and send them to the signal generator. Every HDLC packet contains a serial number which is periodically incremented. This makes it possible to count the number of received and lost packets. The software used to calculate the HDLC packet bits was written with LabVIEW from National Instruments[2] and communicates with the signal generator via the GPIB interface.

Signal generator

The signal generator is used to output a waveform containing the HDLC bits, and repeat it periodically.

Modulator

The modulator modulates the carrier with the waveform and generates a RF signal that can be received with the AX5042.

AX5042 /uC

The AX5042 together with the microcontroller receives the modulated waveform and counts the number of packets received and lost. With this, the PER can be easily calculated. The signal strength of the modulator is now slowly reduced until the limit of 10%PER (complies to 0.1% BER) is reached.

For 868MHz and 4.8kbps the expected sensitivity for 10% PER is at -114dBm (according to AX5042 data sheet).

To test if the frequency acquisition is working, increase the signal level to a value where the receiver is running fluently and then increase / reduce the carrier frequency by changing the modulator settings. The receiver should now be able to receive the signal even with an increased frequency offset.

Performance testing and tuning

The sensitivity may not be as expected. There are many reasons for this and mostly it is because of bad register settings. The following list shows several parameters that can be tuned to improve sensitivity.

preamble length

The preamble length when transmitting a packet must be long enough to make sure the receiver has enough time to perform frequency acquisition before the packet data starts. Use the value calculated with the excel sheet as a starting point and slightly increase or decrease the number of preambles until optimal sensitivity is reached.

AGCATTACK, AGCDECAY

These registers define the speed of the automatic gain control. Setting the registers to a low value will require longer time until the AGC is adjusted, but will result in a better PER.

FREQGAIN

The default value for FREQGAIN is 0xa, but this must not be the optimal value. Start with the calculated value and modify the FREQGAIN register to see if there is any improvement in the sensitivity.

deviation

In some situations increasing the transmitter deviation improves receiver sensitivity especially when the channel filter bandwidth is at its minimum of 4.8kHz.

IFFREQ, Intermediate frequency

Not all values for the intermediate frequency are suitable and setting the IFFREQ register to a value other than 1.000MHz is strongly recommended.

limitations

There are some limitations however. The bandwidth of the receiver cannot be reduced to any value, there is a lower limit and even when reducing the bit rate further it won't bring any improvement in sensitivity. The lower limit for the AX5042 is reached when the CICDEC register is set to 0x200 (4.8kHz bandwidth, 3.2kbps @ h=0.5). When using bitrates lower than 3.2kbps, increasing the modulation factor h to a value higher than 0.5 might improve sensitivity about 1-2dBm.

2. Appendix

2.1. References

AX5042 downloads

Download AX5042 datasheet and programming manual from <http://www.axsem.com>

Download frequency acquisition source code from <http://www.axsem.com>

[1] AXEVK used to generate code snippets: <http://www.axsem.com>

AX5042 HARDWARE

Find local distributors for AX5042 transceiver at <http://www.axsem.com>

OTHER LINKS

[2] LabVIEW (National Instruments IDE): <http://www.ni.com/labview>

3. Contact Information

AXSEM AG

Oskar-Bider-Strasse 1
CH-8600 Dübendorf
SWITZERLAND

Phone +41 (44) 882 17 07

Fax +41 (44) 882 17 09

Email sales@axsem.com

www.axsem.com

For further product related or sales information please visit our website or contact your local representative.

The specifications in this document are subject to change at AXSEM's discretion. AXSEM assumes no responsibility for any claims or damages arising out of the use of this document, or from the use of products based on this document, including but not limited to claims or damages based on infringement of patents, copyrights or other intellectual property rights. AXSEM makes no warranties, either expressed or implied with respect to the information and specifications contained in this document. AXSEM does not support any applications in connection with lifesupport and commercial aircraft. Performance characteristics listed in this document are estimates only and do not constitute a warranty or guarantee of product performance. The copying, distribution and utilization of this document as well as the communication of its contents to others without expressed authorization is prohibited. Offenders will be held liable for the payment of damages. All rights reserved. Copyright © 2007 AXSEM AG