NXP Semiconductors
Application Note

Document Number: AN5300
Rev. 0, 07/2016

Using and Synchronizing the KEA's Internal Clock for LIN Slave Implementations

by: Yves Briant and Ricardo Duran

1 Introduction

Reliable communication via the asynchronous LIN protocol requires an MCU with a bus clock accurate enough to avoid errors. MCUs that use clocks based on crystal or ceramic resonators easily provide very accurate bus clocks. The LIN protocol was designed to also allow more cost-effective solutions: MCUs with on-chip oscillators can be used successfully to implement LIN slaves, even though the on-chip oscillators have less accuracy than a crystal.

A LIN master initiates each frame by sending a 13 bit break field and a synchronization byte. The data of the synchronization byte is always 0x55.

Thus this byte is composed of five falling (recessive to dominant) edges that can be used as a reference for clock and/or baud rate adjustment.

Each derivative of the KEA's family embeds an internal oscillator, whose output frequency can be changed by adjusting a trim register. This application note proposes a method to adjust the frequency of NXP's KEA internal oscillator by measuring the synchronization byte sent within a LIN frame and adjusting the corresponding trim register in order to be within the tolerance range allowed for LIN slave nodes.

Contents

1 Introduction....1
2 Baudrate requirements for LIN....2
3 KEA's internal clock source....2
4 Implementation of the internal clock synchronization....3
4.1 Correction of the internal clock 4
4.1.1 Computation of the new trim value....4
4.1.2 Clock synchronization algorithm....6
4.1.3 Numerical example....8
4.2 Remarks....9
5 Synchronization performance....10
5.1 Accuracy of the synchronization....10
5.2 CPU loading....11
5.3 Settling time....11
6 Conclusion....11
7 Example of implementation....11
8 References....12

2 Baudrate requirements for LIN

The required clock accuracy is different for a LIN slave and a LIN master. These requirements are summarized in the table below:

Table 1. Bit rate tolerances relative to nominal bit rate

No.	Bit rate tolerance	Name	$\Delta F/F_{Nom}$
Param 1	Master node (deviation from nominal bit rate)	$F_{TOL\_RES\_MASTER}$	<±0.5%
Param 2	Slave node without making use of synchronization (deviation from nominal rate)	$F_{TOL\_RES\_SLAVE}$	<±1.5%
Param 3	Deviation of slave node bit rate from the nominal bit rate before synchronization; relevant for nodes making use of synchronization and direct break detection.	F_{TOL_UNSYNC}	<±14%

Since a LIN master initiates each LIN frame and sends the break and the synch fields, it should embed an accurate clock source (<0.5%).

For a LIN slave, two different accuracies are required: < ±14% for the reception of the break and synchronization fields, <1.5% for the reception or emission of the remainder of the LIN frame. These two tolerances are illustrated in the figure below.

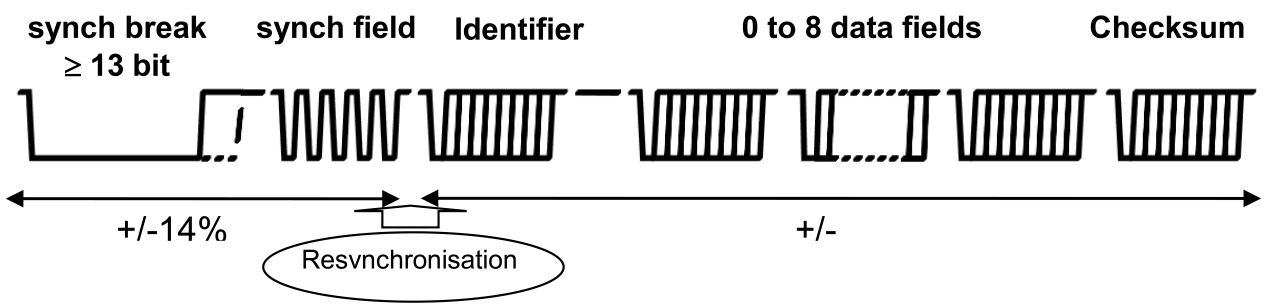

Figure 1. Baudrate tolerances for a LIN slave

The requirement to have a ±14% accurate clock source to reliably recognize the break signal is achieved by the factory trimming of the oscillator on all of the KEA devices. To achieve the more precise ±1.5% requirement for LIN communication, a further adjustment is required. These adjustments are covered in the next two sections.

3 KEA's internal clock source

The internal clock module within the KEA family:

- ICS (Internal Clock Source): The main clock source generator providing bus clock

The internal clock source (ICS) module provides clock source choices for the MCU, with the limitation of its accuracy being in the range of -2.3% (min) to 0.8% (max) as specified in the datasheet. The module contains a frequency-locked loop (FLL) as a clock source that is controllable by either an internal or an external reference clock.

This Application Note will only focus on the internal reference clock. For more details about these clock modules, go to the device's reference manual and datasheet at: Reference Manual & Datasheet.

The internal reference clock behaves the same way as with other microcontrollers integrating an internal oscillator: their output frequencies vary with the temperature, supply voltage and process.

The largest contributor to this frequency deviation is the process variation (±20%). Since this deviation is fixed and does not vary with the time, it can be entirely cancelled by one single initial (in factory) trim operation.

The frequency deviation due to temperature is guaranteed to be less than ±4% over -40°C to 125°C, whereas the effect of the supply voltage variation can almost be negligible, as long as the supply voltage remains within 5% of the nominal voltage. These two parameters (temperature and supply voltage) vary with the time and their variations should be compensated to assure that the internal clock frequency remains stable. The frequency variations can be cancelled by adjusting a "trim register". This trim register can change the output frequency by ±25%, with a resolution finer than 0.4%.

The internal clocks are factory trimmed: a value of the trim register is determined and recorded into flash memory. The frequency deviation due to the process variation can be almost entirely eliminated. Taking in to account the temperature, the supply voltage and the remaining process variations, the overall accuracy of the factory trimmed internal reference is much better than ±14%, which means that the break and synch fields of the incoming frame can be correctly received without any initial synchronization.

The table below summarizes the main characteristics of KEA's internal clock module:

Table 2. Main characteristics of the KEA's internal clock

	Frequency range - untrimmed -	Frequency range - factory trimmed-	Frequency deviation with temperature and supply	Trim register	Trim resolution 8bits	Trim resolution 9bits
ICS KEA (128 & 8)	[31.25 ; 39.0625] ± 25%	37.5 kHz ± .8%	Max ±1.5%	9 bit	Max: ± 0.4%	Max: ± 0.2%
					Typ: ±0.2%	Typ: ±0.1%
ICS KEA64	[31.25 ; 39.0625] ± 25%	31.25 kHz ± .8%	Max ±1.5%	9 bit	Max: ± 0.4%	Max: ± 0.2%
					Typ: ±0.2%	Typ: ±0.1%

Temperature range: [-40; + 125]°C
Supply range: 5V± 10%
All parameters min and max values

4 Implementation of the internal clock synchronization

The implementation described in the following section applies for all KEA derivatives..

- The synch field, sent by the LIN master is used as a fixed reference clock (in fact a ±0.5% accurate clock signal). The duration of this synch field can be measured by any LIN slave, using a timer channel. Since the KEA doesn't implement an internal connection between the SCI's RX and a timer channel, an external connection is required (as seen on Figure 5, later in the document)
- Since the timer channel is clocked by the internal oscillator of the slave, the measured duration of the synch field illustrates the clocks deviation from the internal oscillator's expected frequency. If the measured duration is shorter than the expected one (assuming the nominal frequency of the internal oscillator), it means that the internal oscillator is running too fast, and vice-versa.
- From this measured duration, a new trim value can be computed and loaded into the trim register of the internal oscillator. If the internal clock oscillates too fast, the trim value should be increased, and vice-versa.

Implementation of the internal clock synchronization

Once the process variations have been cancelled out by the initial trim, the major contributor to the internal clock deviation is the temperature. Since temperature is a slow changing parameter the system may be able to cope with re-synchronizing the clock less often to save CPU bandwidth, however this strategy must be assessed against the improbable, but possible effect of communication failure.

The diagram below shows how the clock re-synchronization is achieved:

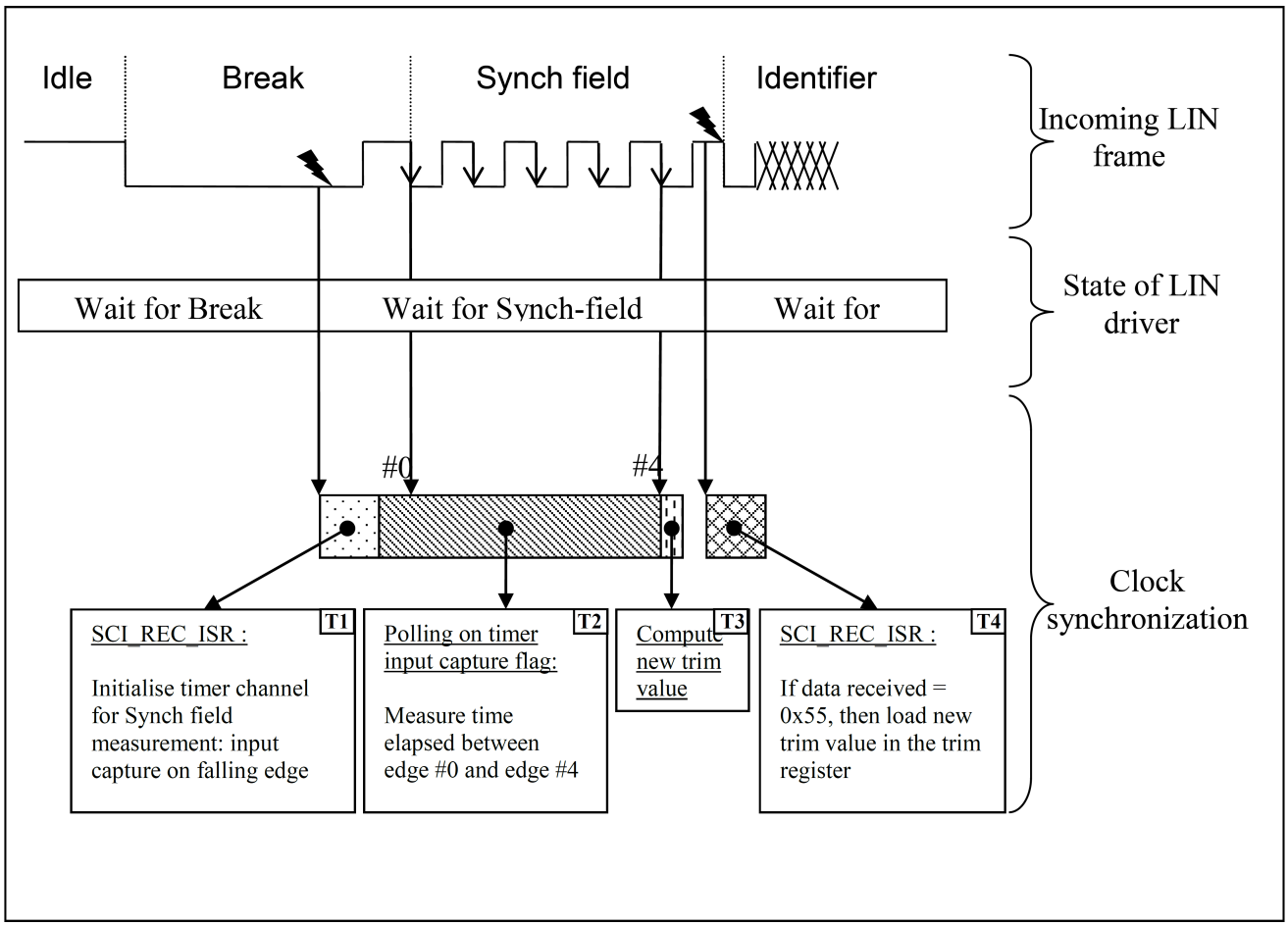

Figure 2. Principle of the clock synchronization

The measurement of the synch field duration is achieved by polling the input capture flag. The timer need to have the appropriate priority level in order to allow successful measurement.

4.1 Correction of the internal clock

4.1.1 Computation of the new trim value

The duration T_{synch} of the synch-field is measured between the first and the fifth falling edge, corresponding to 8 bits. Therefore:

$$T_{synch} = 8Tbits = \frac{8}{LIN\_baudrate} \quad (1)$$

Where *LIN_baudrate* is the nominal baudrate on the LIN network, assuming no clock deviation.

With the SCI module embedded in the KEA, the baudrate is generated from the bus frequency, according to the following formula:

$$LIN\_baudrate = \frac{F_{bus}^{nom}}{16 \times LIN\_prescaler} \quad (2)$$

Where F_{bus}^{nom} is the nominal bus frequency and LIN_prescaler is the 16-bit content of the UARTx_BDL and UARTx_BDH registers.

It yields from (1) that:

$$T_{synch} = \frac{8 \times 16 \times LIN\_prescaler}{F_{bus}^{nom}} = \frac{128 \times LIN\_prescaler}{F_{bus}^{nom}} \quad (3)$$

The measurement of Tsynch with the KEA timer (FTM) results in the digital value V_{meas}, which has following expression:

$$V_{meas} = \frac{T_{synch}}{T_{timer}} \quad (4)$$

where T_{timer} is the period of the clock source of the timer.

The timer should be clocked by the bus frequency, fed by the internal oscillator. With F_{bus}^{act} referring to the "actual bus frequency", it yields:

$$T_{timer} = \frac{timer\_prescaler}{F_{bus}^{act}} \quad (5)$$

From (3), (4) and (5):

$$V_{meas} = \frac{T_{synch}}{T_{timer}} = \frac{128 \times LIN\_prescaler}{F_{bus}^{nom}} \times \frac{F_{bus}^{act}}{timer\_prescaler} = \frac{F_{bus}^{act}}{F_{bus}^{nom}} \times \frac{128 \times LIN\_prescaler}{timer\_prescaler}$$

By defining

$$\frac{128 \times LIN\_prescaler}{timer\_prescaler}$$

as the constant *F_DEV_FACTOR* the internal oscillator deviation can be calculated directly from V_{meas}:

$$\frac{F_{bus}^{act}}{F_{bus}^{nom}} = \frac{V_{meas}}{F\_DEV\_FACTOR} = 1 \pm \Delta \quad (6)$$

It can be observed from this formula, that if V_{meas} is greater than the constant *F_DEV_FACTOR*, it means that the actual bus frequency is greater than the nominal bus frequency, and that the trim value should be increased.

From the equation (5), we can also get that:

$$\Delta_{trim} = \frac{\Delta}{0.4\%} = 250 \times \Delta = \left| V_{meas} - F\_DEV\_FACTOR \right| \times \frac{250}{F\_DEV\_FACTOR}$$

Defining the constant $F\_CORR\_FACTOR = \frac{250}{F\_DEV\_FACTOR}$ this means:

$$\boxed{\Delta_{trim} = \left| V_{meas} - F\_DEV\_FACTOR \right| \times F\_CORR\_FACTOR} \quad \textbf{(7)}$$

4.1.2 Clock synchronization algorithm

The flowcharts below summarize the contents of the task 3 (“compute new trim value”) and of the task 4 (“apply new trim value”) as shown on Figure 3 and Figure 4 below:

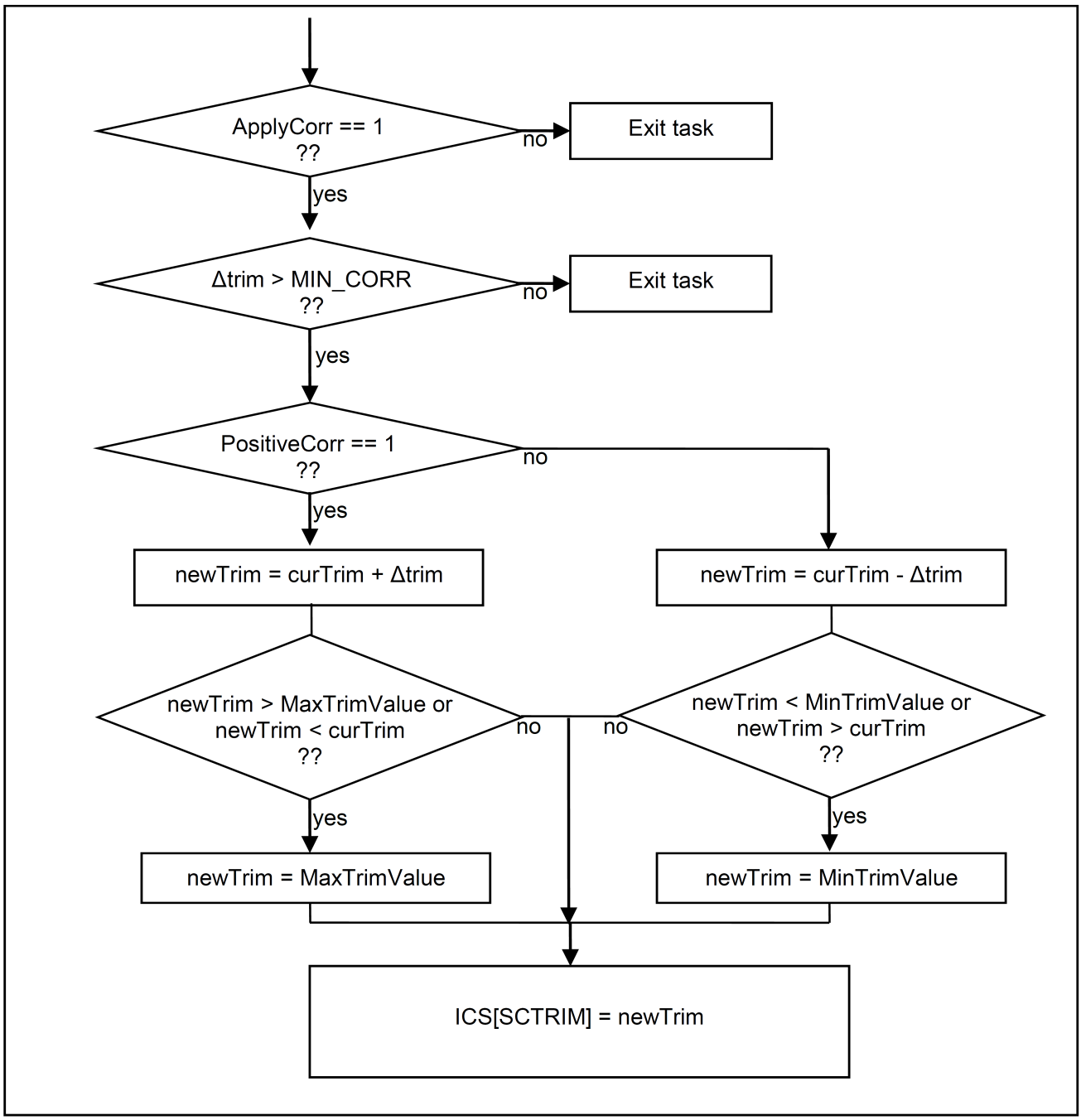

Figure 3. Flowchart of the "Compute new trim value" task

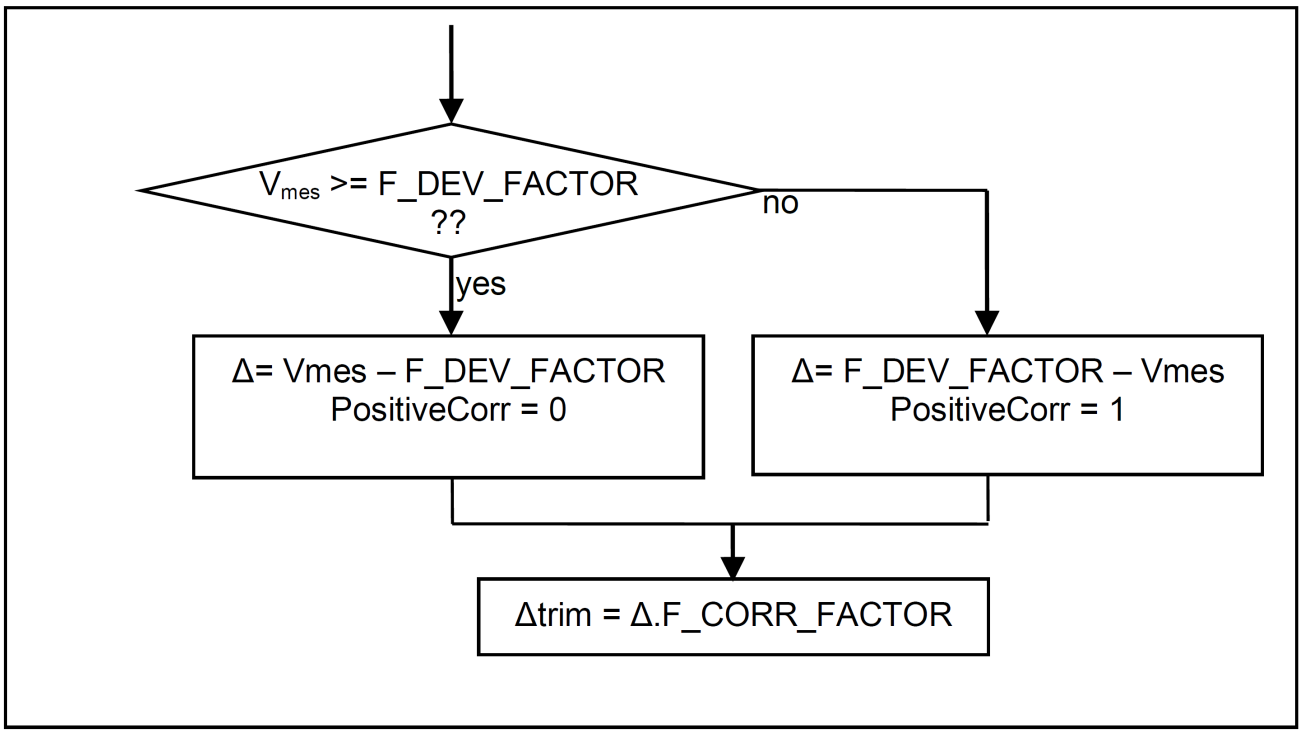

Figure 4. Flowchart of the "Apply new trim value" task

ApplyCorr is a variable regularly updated (for example by the Real-Time-Counter interrupt) that defines the update-rate of the internal clock.

MIN_CORR is a constant defining the granularity of the clock synchronization. Any change of the trim value smaller than *MIN_CORR* will not be applied.

PositiveCorr is a variable updated by the task "Compute new trim value", representing the sign of the trim correction.

MaxTrimValue and MinTrimValue are some constants defining the maximum cumulative correction that could be applied.

4.1.3 Numerical example

Consider the example of a LIN slave implemented with a KEA128 containing an ICS module. This slave communicates on a LIN network at a nominal baudrate of 9600 bauds, and is clocked at a nominal bus frequency Fbus nom =20 MHz, meaning a LIN_prescaler value of 130.

The LIN and Timer prescalers can be calculated using equations (2) and (8):

Timer_prescaler > LIN_prescaler/2=65. Timer_prescaler = 128 is the best choice on the KEA128

Therefore,F_DEV_FACTOR = 128 * 130/ 128 = 130

F_CORR_FACTOR = 250 / 130 = 1.92 which is approximated to 2

Consider the situation where the actual bus frequency is not the nominal 20 Mhz but 19.685 Mhz (20 Mhz – 1%).

$$V_{meas} = \frac{8}{9600} \times \frac{19.685Mhz}{128} = 128$$

therefore, according to the equation (7): Δ_{trim}= (130-128)x1 = 2

The trim value will be increased by 2, which results in a correction of 2x0.4% = 0.8%

4.2 Remarks

1. The synchronization process only works if the initial trim value (determined at ambient temperature) is far enough from 0x00 and 0xFF to allow at least a ±2% correction of the temperature. Assuming a 0.2% resolution of the trim register, the initial trim value should be greater than 10 (2%/0.2%) and less than 245 (255-2%/0.2%). The factory-trimmed values programmed by NXP are within these two limits.
2. The new trim value should be computed in a way that does not lead to over-correction. There are several reasons for that:
   - The main cause of the internal clock deviation is the temperature, which varies slowly.
   - The internal clock feeds (directly or) the bus frequency, which clocks the peripherals of the LIN slave. Having a "restrained" clock correction avoids any over-oscillation of the bus frequency.
   - Most of the time, the internal clock is the reference of the FLL whose output feeds the bus frequency. If the internal clock correction is too big, there is a small risk that the FLL could lose lock, leading to loss of communication on the LIN network.
3. The resolution of the trim process is maximum 0.4% (see table 2) but no minimum value guaranteed. This means that the computed correction (which assumes 0.4% in the example above) may not have the desired impact and be too small to compensate for the frequency deviation, therefore two or three correction cycles will be required to accommodate for the frequency deviation.
4. For this application note, an external hardware connection is necessary because KEA can't map the necessary pins internally. The connection by hardware only need one wire to connect between a FTM pin and the UART's RX pin to measure the LIN frame received. Image below shows an example of the pins that could be connected between them. In this particular case, the connection is between pin PTF2 (RX) and PTG7 (timer input capture channel)

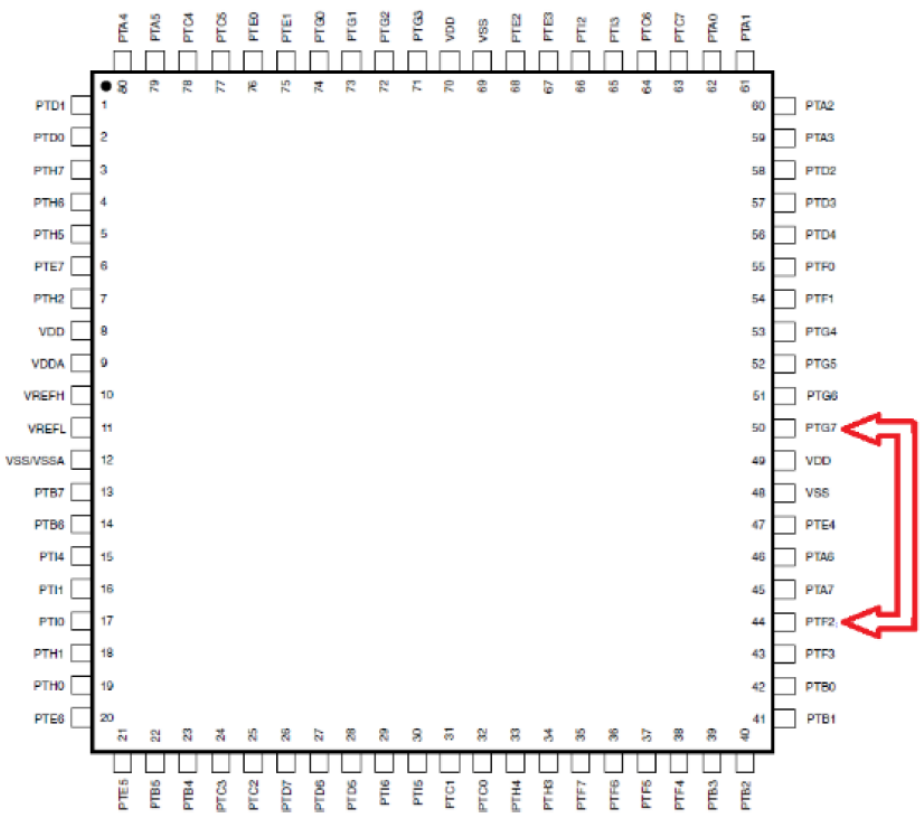

Figure 5. use of FTM2 CH5 (PTG7) on 80pin package to measure the synch field length

5 Synchronization performance

The algorithm described in this application note has been implemented and tested on the S9KEAZ128AMLK using the NXP evaluation board “FRDM-KEA128” using CodeWarrior for Microcontrollers. Both the accuracy and the CPU load required by this algorithm have been measured.

5.1 Accuracy of the synchronization

The accuracy of the synchronization has been tested for a deviation in the range [-3%;3%] . The nominal baudrate is 9600 bauds and the bus frequency is normally 20 MHz.

The results are the following:

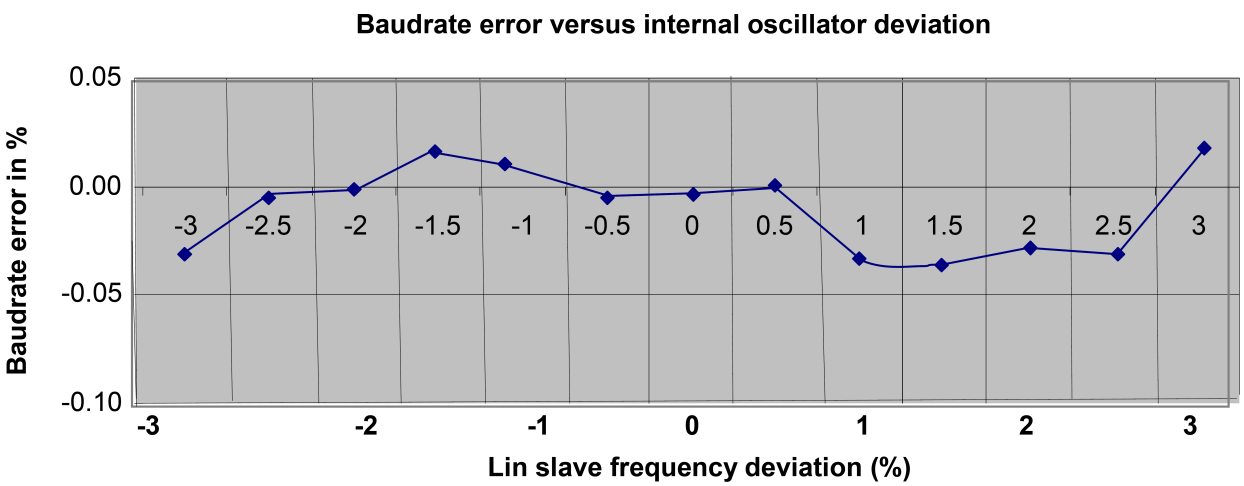

Figure 6. Accuracy of the frequency correction on a KEA128

This figure shows the accuracy of the synchronization is always better than 0.1%, which is far better than the LIN communication requirement of 1.5%.

Several observations can be made on the achieved accuracy:

- The measurement of the synch-field with the 8bit timer introduces an error Em:

  Em < 1/(256 x 2) = 0.2%

- The accuracy of one synchronization depends on the height of a trim step (the frequency deviation caused by an increment of the trim value), which is in the less than or equal to 0.4%. It also depends on the approximation when computing the integer constant F_DEV_FACTOR and F_CORR_FACTOR. These two factors lead to an underestimation of the correction, which is the recommended behavior.
- This synchronization method is a closed loop control system because the effect of one correction affects the next measured synch-field. The accuracy of one correction is not as relevant as other criteria, like the stability and the damping of the response.

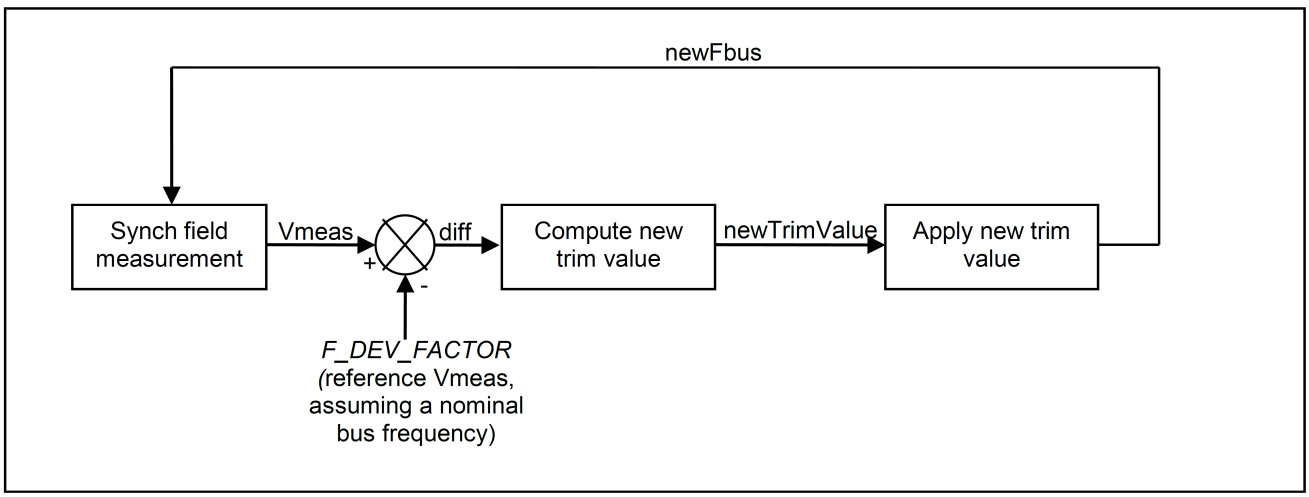

Figure 7. Synchronization method shown as a feed-backed control system

5.2 CPU loading

When this method is used (principle of the synchronization, the CPU is loaded at 100% only during the reception of the first 8 bits of the synch field. After this measurement, the computation of the new trim value (reference Figure 2) requires 4.03 µsec with the KEA running at 20 MHz.

5.3 Settling time

The internal clock of the KEA can be selected to feed the Frequency Lock Loop (FLL), allowing a high bus frequency. When changing the trim register, the reference frequency of the FLL changes, and the FLL may loose its lock, yielding in larger frequency deviations until it locks again.

Since the corrections of the trim value are very slight (and can be limited by software) the FLL is not at risk of loosing lock. In practice, the FLL will not loose lock with frequency changes less than 2%.

6 Conclusion

The requirement to have a ±14% accurate clock source to reliably recognize the break signal is achieved by the factory trimming of the oscillator on all of the KEA devices. To achieve the more precise ±1.5% requirement for LIN communication, further adjustment is required. This previously described method of synchronization exhibits accuracy far beyond the LIN protocol requirements for a slave node, with the limited impact of some additional CPU load, one additional timer channel and external connection on the MCU.

7 Example of implementation

This project has been migrated from the app note for the S08SG32 (app note: AN3756) available from www.nxp.com

KEA implementation and right steps to integrate it into an application:

First point toconsider when implementing the Re-synch algorithm:

- LIN baud rate as mentioned in Computation of the new trim value.
- Internal clock trimming mechanism.
- Then compute the maximum and minimum allowed deviation trim value,

Use the following three steps to apply the RE-synch algorithm, initial trim clock configuration, obtain maximum and minimum allowed trim value and finally the break detection and the computing of the new trimming value.

Step 1:

In the function main():

After the peripheral initialization, a call to the function “initTimer4LIN()” is added for the configuration of the channel 5 of the timer 2 for the synch field measurement. Note: Do not forget the hardware connection UART_RX to Flextimer.

Step 2:

The function “initTrimSaturation()” is called to compute the maximum and minimum allowable trim value, depending on the maximum amount of the clock correction that the user wants to allow.

Step 3:

In the interrupt routine UART1_SCI1_IRQHandler ():

When a break character has been received, the measurement of the synch field is armed. The polling method is used to measure the length of the synch field. After the fifth falling edge, the measurement is stopped and the value of the clock correction is computed.

NOTE

When a synch character has been received, the clock correction is applied if the variable “ApplyCorr” is set. In this implementation, ApplyCorr is set by RTC interrupt each 10 msec.

The example SW is provided separately as a reference.

8 References

AN3756 - Using and Synchronizing the S08’s Internal Clock for LIN Slave Implementations.

KEA128 Reference Manual

LIN Specifications

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer’s technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTest, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and µVision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2016 NXP B.V.

Document Number AN5300
Revision 0, 07/2016