Application Note: JN-AN-1186

JN516x Temperature-Dependent Operating Guidelines

When using the JN516x device in environments with significant temperature variation, periodic recalibration of the radio is recommended. In addition to this, the 32MHz crystal oscillator temperature characteristic needs to be compensated for operation above 90 ⁰C. This Application Note describes the software functions that perform these operations.

1 Introduction

The JN516x devices feature an integrated radio, which is calibrated at start-up for optimum performance. In operating environments with a significant variation in temperature (e.g. greater than 20°C) due to diurnal or ambient temperature variation, it is recommended to recalibrate the radio to maintain performance. Recalibration is only required on Routers and End Devices that never sleep. End Devices that sleep when idle are automatically recalibrated when they wake.

In order to maintain the tolerance of the radio reference frequency to within the 40ppm limit specified by the IEEE 802.15.4 standard, it is necessary to compensate the 32MHz crystal oscillator at operating temperatures above 90 ⁰C. All AT cut quartz crystals exhibit a steep rise in resonant frequency above 90-100 ⁰C. Therefore, provision has been made in the 32MHz crystal oscillator of the JN516x to increase the crystal load capacitance and thereby reduce the oscillating frequency – this process is known as 'frequency pulling'.

This Application Note describes a set of software functions which measure temperature using the on-chip temperature sensor and trigger a recalibration if there has been a significant temperature change since the previous calibration. Frequency pulling of the main crystal oscillator by adjusting the crystal load capacitance can be performed using a standard integrated peripheral API function.

2 Requirement for Periodic Recalibration

The JN516x radio has been designed so that on start-up it selects a calibration curve for the voltage controlled oscillator (VCO) to produce the specified performance from the devices. If the operating/ambient temperature of the device changes significantly from when calibration occurred, the device performance may be reduced. Recalibration is then required to maintain performance.

The recalibration process may be triggered under software control. Due to the time taken (less than 200 μs) to recalibrate the radio, there is the potential for packets to be missed on heavily loaded nodes (e.g. Routers).

Bench testing has shown a small increase in PER when operating at the limits of the calibration curve. However, process variation during device manufacture may lead to a more significant effect on some devices, so periodic recalibration is recommended.
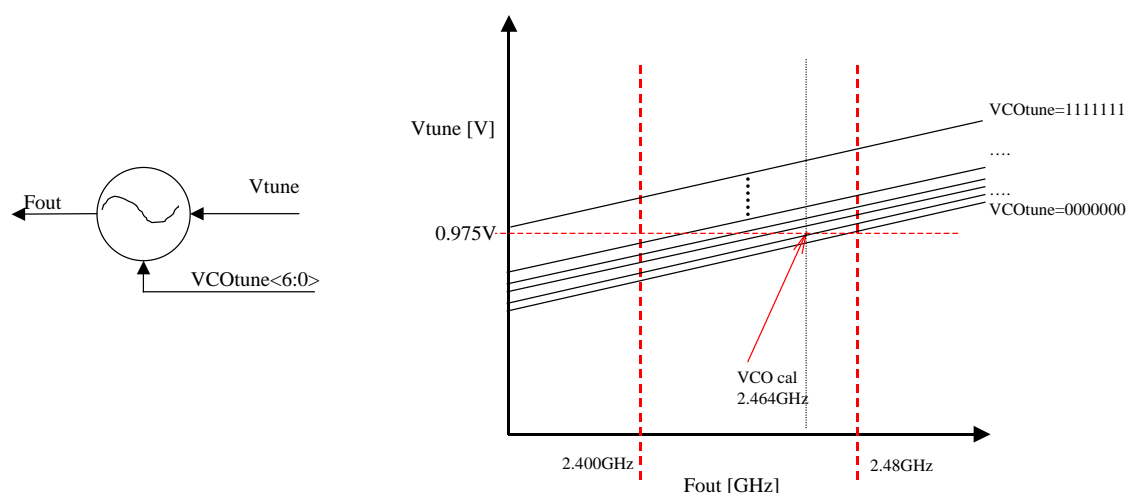
2.1 Theoretical Behaviour

Figure 1: Simplified Diagram of VCO and Characteristics

Figure 1 shows an idealised VCO behaviour on the JN516x device. It has an input voltage, *Vtune*, a 7-bit tuning word, *VCOtune<6:0>*, and an output sinusoid port.

The *VCOtune* parameter is chosen to ensure that the VCO can meet the tuning range across all channels from 2.403GHz (RX lowest channel) to 2.48GHz (TX highest channel). In addition, the VCO must operate with a tuning voltage between 0.7V and 1.25V to meet the PLL design specifications. Figure 1 shows that out of the 127 possible tuning codes, no one code covers the whole frequency band. Process variation may lead to a faster or slower VCO, which may effectively raise or lower the *VCOtune* lines in the above graph. In this case, one of the other *VCOtune* values will then be selected. Temperature variation will also change the device operation, which may lead to the F_{out} falling outside of the specified operating region, in which case an alternative *VCOtune* line would be selected by recalibrating.

2.2 Expected Behaviour

The effects and side effects of operating the radio outside of its calibrated range may theoretically include the transmitter operating 'off-channel' and a reduction in receive sensitivity.

JN-AN-1186 (v1.0) 18-Apr-2013

If the radio is outside of its calibrated range, the transmitted power will be maintained but may be off-channel and even out-of-band (greater than 2.48GHz or less than 2.405GHz.). The receiver sensitivity may be degraded to the point where no packets are received, i.e. PER is 100%.

During our tests, 'worst case' silicon (due to process variation) has not been identified that shows this behaviour. However, when the VCO was intentionally set to 'uncalibrated', it resulted in 100% PER and no on-channel transmitted signal, until recalibration was performed. When the radio was re-calibrated, normal operation was restored.

3 Requirement for 32MHz Crystal Oscillator Correction

All AT cut quartz crystals exhibit a steep rise in resonant frequency above 90-100 ⁰C, therefore, to keep the VCO frequency with the 40ppm limit set by the IEEE 802.15.4 standard, it is necessary to provide a means to compensate for this rise in frequency. This is achieved by decreasing (pulling) the frequency of the crystal by increasing the load capacitance seen by the crystal. Extra internal capacitors of about 1pF can be switched in on each of the two oscillator pins (XTAL_OUT and XTAL_IN). The 3 states are; no additional capacitance, which is the default state (code 0), 1pF extra load capacitance (code 1 or 2) and 2pF extra load capacitance (code3). The figure below shows how adding the capacitance avoids exceeding the frequency error limit at high temperatures.
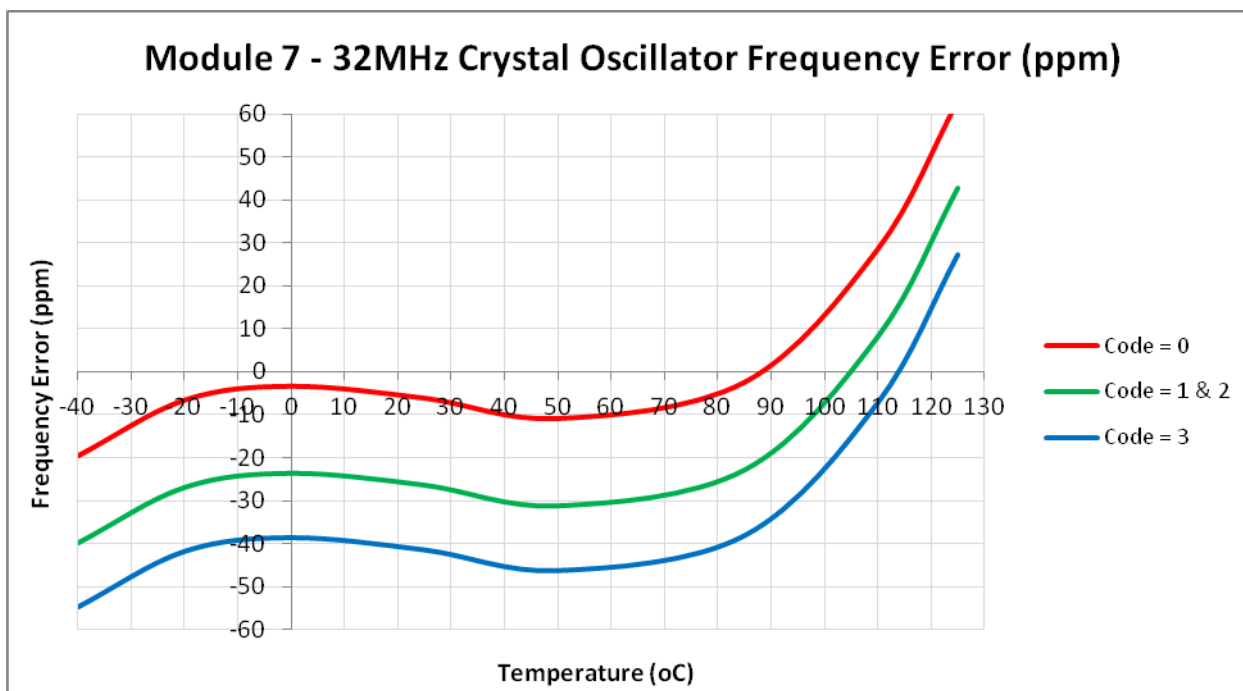

Figure 2 The effect on the frequency error of adding load capacitance

In a typical application for high temperature operation, two thresholds would be set, for example 95 ⁰C and 110 ⁰C. So at 95 ⁰C the frequency error would move from the red trace to the green and then at the higher threshold to the blue trace.

When selecting a crystal suitable for operation up to 125 ⁰C, it is important to check the value of the pulling coefficient. The crystal pulling coefficient specifies the sensitivity of the crystal frequency with respect to the crystal load capacitance. Crystals suitable for use with the JN516x will typically have a crystal pulling coefficient value in the range of 15 to 25 ppm/pF. Although the crystal pulling coefficient has a positive value, it should be noted that the crystal frequency will decrease with increasing crystal load capacitance.

The formula for calculating the crystal pulling coefficient (Δf) is given by:

$$\Delta f = \frac{C_m \times 10^6}{2 \times (C_L + C_S)^2} \, ppm/pF$$

where,

C_m is the crystal motional capacitance (e.g. 4.4fF)
C_L is the crystal load capacitance (e.g. 9pF)
C_S is the crystal shunt or package capacitance (e.g. 1pF)

The example crystal capacitance values quoted above yield a crystal pulling coefficient of 22ppm/pF. Therefore, an increase of the crystal load capacitance (C_L) by 1pF will reduce the crystal oscillating frequency by 22ppm.

In section B4 of the JN516x IEEE802.15.4 Wireless Microcontroller datasheet, the following crystals are listed, which have been tested and shown to perform well with regard to frequency pulling, but the user is not limited to these. AEL X32M000000S039 and Epson Toyocom X1E00002101670 (CL = 9pF, Max ESR 40R).

See section 5.2 for details of adjusting the crystal load capacitance.

4 Application Scenarios

Due to the wide variation in applications for the JN516x devices, it is desirable for the developer to identify how rapidly the temperature may vary in the application operating environment. The causes of temperature variation may range from slow diurnal (Day/Night) changes (e.g. equipment located outdoors) to rapid changes when co-located equipment is switched on (e.g. lighting fixture). The rate of application data transmission and loading on the network may also be taken into account when determining the checking period for the calibration.

Once the temperature range and rate of change of temperature have been identified, the time-period for checking the calibration may be estimated. This time-period should be chosen to be less than the time taken for the ambient temperature of the chip to change by the recalibration range. For example, for an environment with diurnal temperature variation of 80°C, the time taken for the temperature to change by this amount will be ~12 hours – that is, changing by 20°C in 3 hours, and so the recalibration check-time period would be set to 3 hours. In practice, a more frequent call may be made.

In lighting applications, the lamp fixtures can reach temperatures well above 100 deg C, and therefore 'pulling' of the main crystal oscillator will be required. The application program will know when the lamp is turned on or off, or if the lamp has a dimmer function, when the light level is changed. In these circumstances the temperature sensor should be checked more frequently, then when the lamp is in a steady state. The aim should be to guarantee, that a change of 2 deg C can be detected. The sampling time selected will depend on the thermal inertia of the fixture and the power consumed by the lamp.

5 Software Functions

5.1 Recalibration Functions

To maintain device performance and to enable an application to control when recalibration takes place, the following user callable functions are available:

bAHI_PeriodicRecalInit()

bAHI_PeriodicRecal()

bAHI_PeriodicRecalRange()

eAHI_AttemptCalibration()

Recalibration can be performed semi-automatically by setting up and using the periodic recalibration module. Alternatively, the application can take complete control of when recalibration is performed.

To enable the periodic recalibration module, a call is made to the **bAHI_PeriodicRecalInit()** function at the start of the application. The calibration period and temperature range can be set, if desired, otherwise the default values of one hour and 20°C will be used. The application then periodically calls the **bAHI_PeriodicRecal()** function to check whether a recalibration is required and, if necessary, recalibrates.

Note that for applications that use the ADC, there is a risk of contention with these functions, as the internal temperature measurement performed also uses the ADC. There is also a risk of the timer expiring, causing this function to be invoked when the application is sending or has just sent data. To minimise this, the application controls when **bAHI_PeriodicRecal()** is called. The application selects a suitable time (e.g. the ADC is not being used and the application is not in the middle of a data transfer) for the recalibration to be performed. This function will only return TRUE if the module deemed recalibration necessary. A return of FALSE does not indicate an error but indicates that recalibration was not performed at this time.

Applications that make extensive use of the ADC and cannot tolerate sharing this resource must perform the temperature measurements in the application and decide when a recalibration is necessary. The function **eAHI_AttemptCalibration()** is provided which will force an immediate recalibration without using the ADC.

Note: It is not necessary to call any of the periodic functions if this direct approach is to be used.

The function prototypes are detailed below.

bAHI_PeriodicRecalInit()

The function initialises the recalibration module:

bool_t bAHI_PeriodicRecalInit(uint32 *u32RecalCheckPeriod***, uint8** *u8TempDelta***);**

where:

- *u32RecalCheckPeriod* is the time (in seconds) between recalibration checks (0– 68719). A value of zero will force the use of the default time of 1 hour. The maximum value gives a period of approximately 19 hours.

- *u8TempDelta* is the temperature change (in degrees Celsius) required before a recalibration is performed (0–40). A value of zero will force the use of the default 20°C.

Returns: TRUE indicates recalibration enabled, FALSE indicates invalid parameter.

This function must be called to enable the periodic timer within the module.

> (i) **Note:** This function must also be called following a MAC reset. JenNet users must catch the stack reset event and call this function to re-enable the periodic recalibration, if required.

bAHI_PeriodicRecalRange()

This function allows the recalibration parameters to be changed on-the-fly:

bool_t bAHI_PeriodicRecalRange(uint32 *u32RecalCheckPeriod***, uint8** *u8TempDelta***);**

where:

- *u32RecalCheckPeriod* is the time (in seconds) between recalibration checks
- *u8TempDelta* is the difference in temperature (in degrees Celsius) before a recalibration is triggered

Returns: TRUE indicates parameters successfully updated. FALSE indicates invalid parameters.

bAHI_PeriodicRecal()

This function requests a periodic recalibration (which is only performed if a recalibration is due):

bool_t bAHI_PeriodicRecal(void);

Returns: TRUE indicates the recalibration module performed a calibration. FALSE indicates that a recalibration was not necessary.

> (i) **Note:** This function must be called in a timely manner to allow the recalibration to be performed.

eAHI_AttemptCalibration()

This function forces the recalibration module to perform an immediate calibration (note that the ADC and internal temperature sensor will not be accessed):

eAHI_AttemptCalibration(void);

Returns: E_CAL_SUCCESS (0x00)

                    E_CAL_SCAN_IN_PROGRESS (0x01)

5.2 Crystal Frequency Pulling Function

The JN516x Integrated Peripherals API contains a function for pulling the 32MHz crystal frequency by the application of additional crystal load capacitance.

          JN-AN-1186 (v1.0) 18-Apr-2013

vAHI_ClockXtalPull()

The function applies additional crystal load capacitance to the 32MHz crystal oscillator.

void vAHI_ClockXtalPull(uint8 *u8PullValue***);**

For further details, please refer to *JN-UG-3087 JN516x Integrated Peripherals API User Guide*.

> (i) **Note:** It is the responsibility of the application to perform a temperature measurement and call **vAHI_ClockXtalPull()** with the appropriate parameter value.

As previously described in section 3 and with reference to Figure 2, a typical application for high temperature operation will have two temperature thresholds, for example 95 ⁰C and 110 ⁰C. At 95 ⁰C, **vAHI_ClockXtalPull()** should be called with a parameter value of 1 and at 110 ⁰C it should be called with a parameter value of 3. A hysteresis of 2 ⁰C below these temperature thresholds should be adequate to avoid unnecessary changing of the crystal load capacitance due to small temperature fluctuations.

6 Including the Software Functions

6.1 Function Header Files

The recalibration functions are included in the library **libRecal_JN516x.a** which is supplied as part of the JN516x SDK libraries, as is the header file **recal.h**. To access these library functions, you are required to include the header file in the application source code:

1. Add **#include "recal.h"** to the application source code.

The crystal frequency pulling function is included in the Integrated Peripherals API header file **AppHardwareApi.h**, which is contained in the JN516x SDK environment. To include this function:

2. Add **#include "AppHardwareApi.h"** to the application source code.

6.2 Modify the Makefile

Add the following lines to the makefile of the application to specify the locations of the above include files:

```
INCCFLAGS += -I$(COMPONENTS_BASE_DIR)/HardwareApi/Include/

INCCFLAGS += -I$(COMPONENTS_BASE_DIR)/Recal/Include/

LDLIBS += libRecal_${JENNIC_CHIP}.a
```

> (i) **Note:** The exact modifications required depend on the application's makefile.

6.3 Modify the Application

Make the following additions in your application code to implement periodic recalibration:

1. Insert a call to **bPeriodicRecalInit()**, specifying the required recalibration time-period and temperature delta.

2. Insert a call to **bPeriodicRecal()** in the main processing loop. Note that:

    - The call to **bPeriodicRecal()** simply allows the application to indicate that it will permit a recalibration to take place – the call does not force a recalibration.

    - The call will only return TRUE if a successful calibration took place.

Revision History

Version	Notes
1.0	First release

Important Notice

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

All trademarks are the property of their respective owners.