STD-302Z Interface Method

By John Bell

◆ Introduction

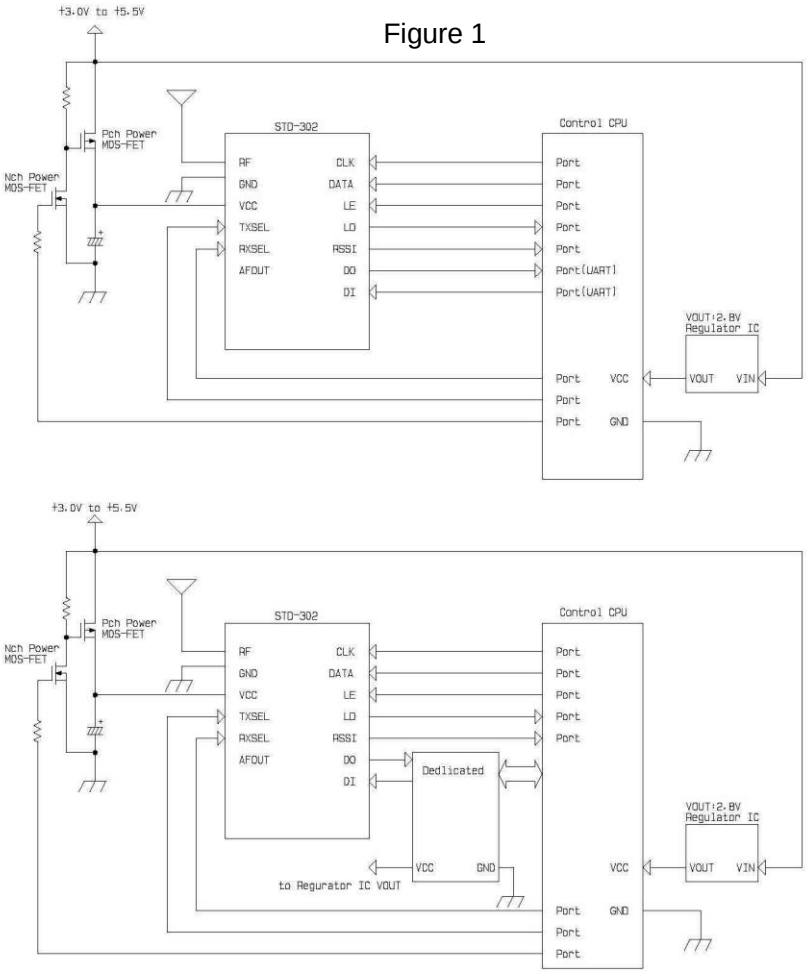
STD-302Z is a module that provides optimized radio function offering flexible control for various industrial applications. It is especially suitable for feedback systems using half duplex communication.
This reference guide explains the interface method for engineers developing applications using STD-302Z.

◆ Typical interface

A typical interface of STD-302Z is shown in Figure 1. The upper example controls the STD-302Z using CPU ports and UART ports, while the lower example uses a dedicated IC with data format or error detection or correction functions. Both systems perform the following processes.

1. The transmission data (digital signal) is input to the DI terminal.
2. The received data (digital signal) is output from the DO terminal.
3. PLL IC control when the RF channel is changed.
4. The lock detect (LD) signal is processed when the RF channel is set.
5. Uses RSSI output.
6. Transmission mode and reception mode control.

Figure 1



◆ Power supply interface

The VCC voltage of the STD-302Z is +3.0 to 5.5 V, however the internal circuit operates at regulated 2.8 V.
Power MOS FET with low ON resistance is used for the power supply interface as shown in Figure 1 to enable ON/OFF control (high active) from the control CPU.
STD-302Z is equipped with an internal PLL frequency synthesizer (Intochips inc. DHL612). The absolute maximum rating for the input terminals of the PLL IC is VCC + 0.5 V, meaning that the absolute maximum rating for the input terminals of the STD-302Z is +3.3 V.
It is recommended that a voltage regulator with 2.8V output is used for power supply of the control CPU. If such a regulator is not used, use STD-302Z within the VCC voltage range of +3.0 to +3.3V.
**If a regulator is not used for power supply of the control CPU, it is recommended that a level conversion circuit is used for the interface to avoid overshoot voltage.*

◆ Input/Output data

Transmission data (digital signal) is input to the DI terminal, which accepts signals up to 4.8 kHz (9800 bps). Received data (digital signal) is output from the DO terminal. However, the internal digitization circuit (slicer circuit) of STD-302Z cannot handle consecutive high or low levels of 10 ms or longer due to its time constant. Therefore it is necessary to devise a method such as processing the data input to the DI transmission terminal so that it does not contain consecutive highs or lows of 10 ms. Data encoding (Manchester coding, CMI coding etc.) is a suitable method, however care should be taken that the frequency bandwidth is not wider than that specified for the module.
If a signal in UART format from the CPU is used, you can transfer data at a maximum of 9600 bps without concern for this issue. For more information about using a UART interface with radio modules, please contact Circuit Design, Inc.

◆ RSSI signal

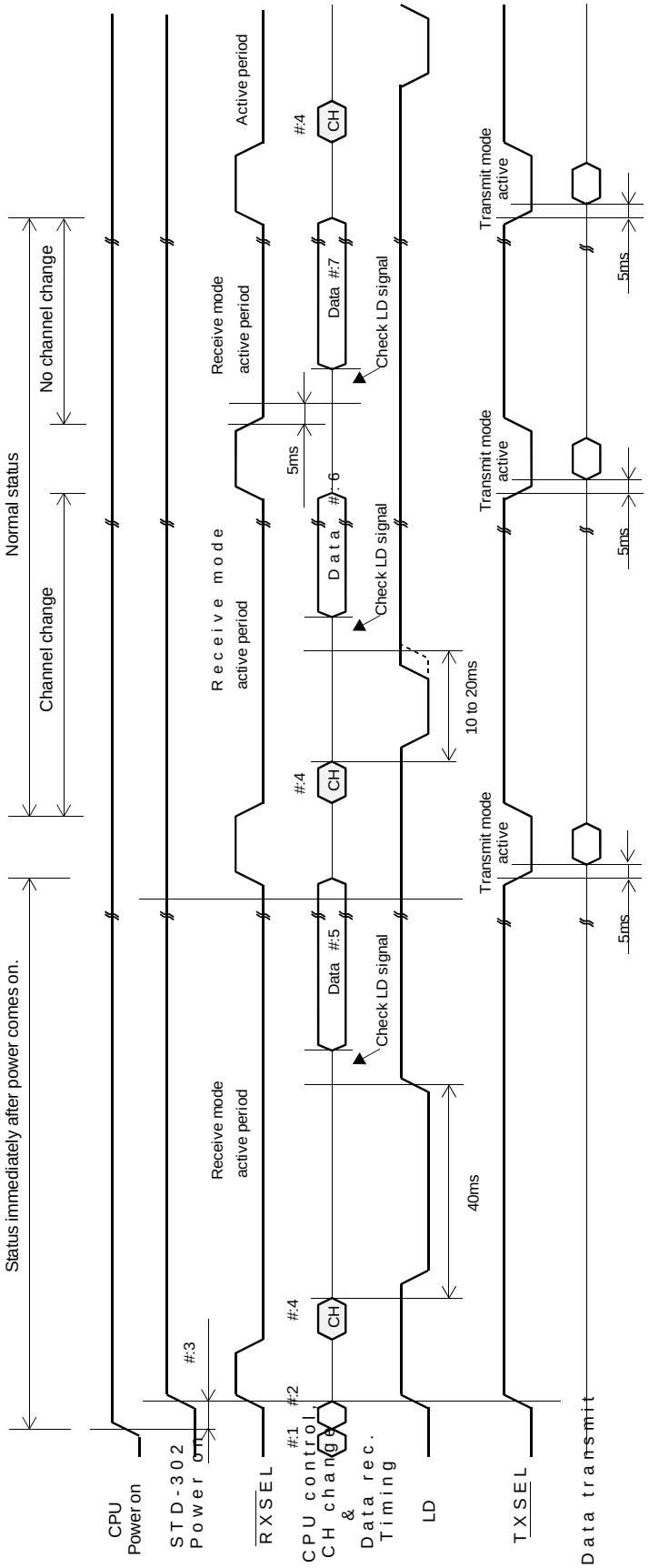
The RSSI terminal outputs the voltage corresponding to the received signal level. This voltage level can be used for functions such as carrier sense[*1], channel auto select[*2], and electric field strength monitoring[*3].
Normally, the CPU processes the RSSI voltage using an A/D converter.
[*1] Carrier sense: A function that judges whether a RF channel is currently open for transmission or not.
[*2] Channel auto select: A function that automatically selects a frequency (RF channel) that is not currently used.
[*3] Electric field strength monitoring: A function that measures the strength of electric field.

◆ Control timing

Control timing in a typical application is shown in Figure 2.
Initial setting of the CPU port to be used is performed when power is supplied by the control CPU and reset is completed. MOS-FET for supply voltage control of the STD-302Z, RXSEL and TXSEL are set to inactive to avoid unwanted emissions. The power supply of the STD-302Z is then turned on. When the STD-302Z is turned on, the PLL internal resistor is unstable. Therefore data transmission and reception is possible 40 ms after the frequency setting data is sent to the PLL. For channel setting except when the power is turned on, STD-302Z can handle the data 20 ms after.
**Setting channels must be carried out in the receive mode. If setting is performed in transmission mode, unwanted emission occurs.*
**Sending PLL setting data while the STD-302Z is turned off may cause latch-up. Ensure that interface communication is performed with the power of the STD-302Z turned on.*

If STD-302Z is switched to the receive mode when operating in the same channel (a new PLL setting is not necessary), it can receive data after 5 ms elapses. For data transmission, if the RF channel to be used for transmission is set while still in receiving mode, data can be sent at 5 ms after the STD-302Z is switched from reception to transmission.
Check that the LD signal is “high” 20 ms after the channel is changed. In some cases the LD signal becomes unstable before the lock is correctly detected, so it is necessary to note if processing of the signal is interrupted. It is recommended to observe the actual waveform before writing the process program.

Figure 2: Timing Diagram for STD-302

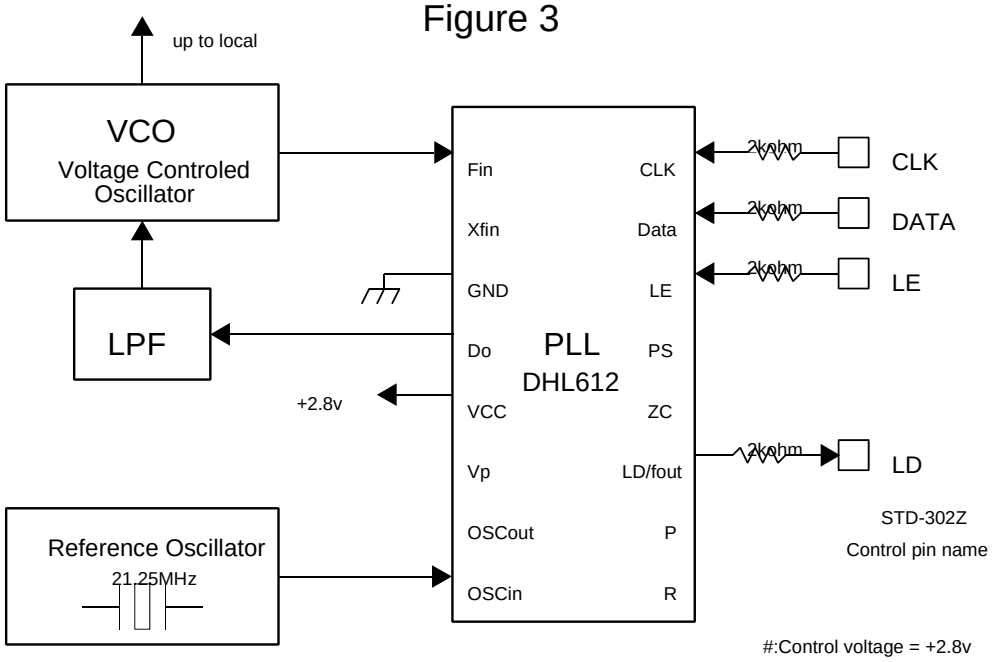


#:1 Reset control CPU
#:2 Initialize the port connected to the module.
#:3 Supply power to the module after initializing CPU.
#:4 RFchannel change must be done in receiving.

#:5 40ms later, the receiver can receive the data after changing the channel..
#:6 10 to 20ms later, the receiver can receive the data after c
#:7 5ms later, the data can be received if the RF channel is not changed.

◆ **PLL IC control**

Figure 3



STD-302Z is equipped with an internal PLL frequency synthesizer as shown in Figure 3. The operation of the PLL circuit enables the VCO to oscillate at a stable frequency. Transmission and reception frequency is set externally by the controlling IC. STD-302Z has control terminals (CLK, LE, DATA) for the PLL IC and the setting data is sent to the internal register serially via the data line. Also STD-302Z has a Lock Detect (LD) terminal that shows the lock status of the frequency. These signal lines are connected directly to the PLL IC through a 2 kΩ resistor.

STD-302Z comes equipped with a Intochips inc. DHL612 PLL IC. Please refer to the manual of the PLL IC. The following is a supplementary description related to operation with STD-302Z. In this description, the same names and terminology as in the PLL IC manual are used, so please read the manual beforehand.

◆ How to calculate the setting value to PLL register

The PLL IC manual shows that the PLL frequency setting value is obtained with the following equation.

$f_{vco} = [(M \times P)+S] \times f_{REF} / R$ -- Equation 1

f_{vco} : Output frequency of external VCO
M: Preset divide ratio of the prescaler (64 or 128)
P: Preset divide ratio of binary 11-bit programmable counter (3 to 2,047)
S: Preset divide ratio of binary 7-bit swallow counter ($0 \leq S \leq 127$ S<P))
f_{REF}: Output frequency of the reference frequency oscillator
R: Preset divide ratio of binary 14-bit programmable reference counter (3 to 16,383)

With STD-302Z, there is an offset frequency (f_{offset}) 21.7MHz for the transmission RF channel frequency f_{ch}. Therefore the expected value of the frequency generated at VCO (f_{expect}) is as below.

$f_{vco} = f_{expect} = f_{ch} - f_{offset}$ ---- Equation 2

The PLL internal circuit compares the phase to the oscillation frequency $f_{vco.}$ This phase comparison frequency (f_{comp}) must be decided. f_{comp} is made by dividing the frequency input to the PLL from the reference frequency oscillator by reference counter R. STD-302Z uses 21.25 MHz for the reference clock $f_{REF.}$ f_{comp} is one of 6.25 kHz, 12.5 kHz or 25 kHz.

The above equation 1 results in the following with n = M x P + S, where "n" is the number for division.

$f_{vco}=n*f_{comp}$ ---- Equation 3 $n = f_{vco}/f_{comp}$ ---- Equation 4 note: $f_{comp} = f_{REF}/R$

Also, this PLL IC operates with the following R, P, S and M relational expressions.

$R=f_{REF}/f_{comp}$ ---- Equation 5 P = INT (n / M) ---- Equation 6 S = n - (M x P) ---- Equation 7
INT: integer portion of a division.

As an example, the setting value of RF channel frequency f_{ch} 433.075 MHz can be calculated as below.
The constant values depend on the electronic circuits of STD-302Z.

Conditions:	Channel center frequency:	f_{ch} = 433.075 MHz
	Constant: Offset frequency:	f_{offset}=21.7 MHz
	Constant: Reference frequency:	f_{REF}=21.25 MHz
	Set 25 kHz for Phase comparison frequency and 64 for Prescaler value M	

The frequency of VCO will be

$f_{vco} = f_{expect} = f_{ch} - f_{offset} = 433.075 - 21.7 = 411.375$ MHz

Dividing value "n" is derived from Equation 4

$n = f_{vco} / f_{comp} = 411.375MHz/25kHz = 16455$

Value "R" of the reference counter is derived from Equation 5.

$R = f_{REF}/f_{comp} = 21.25MHz/25kHz = 850$

Value "P" of the programmable counter is derived from Equation 6.

$P = INT (n/M) = INT(16455/64) = 257$

Value "S" of the swallow counter is derived from Equation 7.

$S = n - (M \times P) = 33921 - 64 \times 530 = 7$

The frequency of STD-302Z is locked at a center frequency f_{ch} by inputting the PLL setting values P, S and R obtained with the above equations as serial data. The above calculations are the same for the other frequencies. Excel sheets that contain automatic calculations for the above equations can be found on our web site (www.circuitdesign.jp/eng/).

The result of the calculations is arranged as a table in the CPU ROM. The table is read by the channel change routine each time the channel is changed, and the data is sent to the PLL.

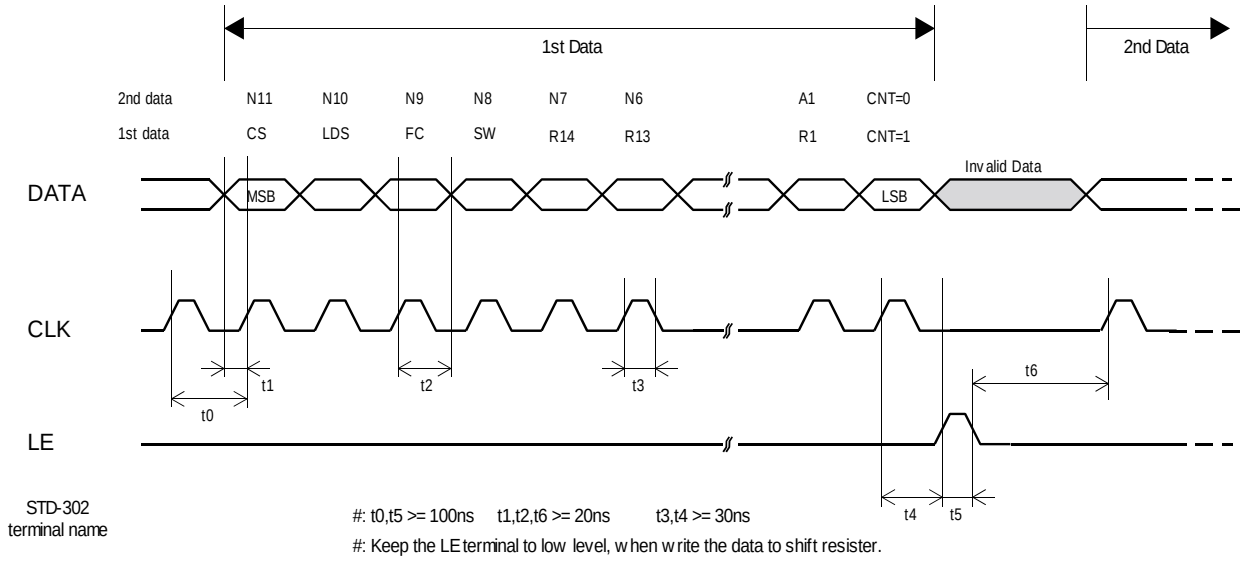
**Hint: You will notice when using the automatic calculation in the Excel sheet, that actually the values R and M can be fixed and only the values A and B are variable.*

◆ Method of serial data input to the PLL

After the RF channel table plan is decided, the data needs to be allocated to the ROM table and read from there or calculated with the software.
Together with this setting data, operation bits that decide operation of the PLL must be sent to the PLL.
The operation bits for setting the PLL are as follows. These values are placed at the head of the reference counter value and are sent to the PLL.

1. CS: Charge pump current select bit
   CS = 0 +/-1.5 mA select VCO is optimized to +/-1.5 mA
2. LDS: LD/fout output setting bit
   LDS = 0 LD select Hardware is set to LD output
3. FC: Phase control bit for the phase comparator
   FC = 1 Hardware operates at this phase

Figure 4



The PLL IC, which operates as shown in the block diagram in the manual, shifts the data to the 19-bit shift register and then transfers it to the respective latch (counter, register) by judging the CNT control bit value input at the end.

1. CLK [Clock]: Data is shifted into the shift register on the rising edge of this clock.
2. LE [Load Enable]: Data in the 19-bit shift register is transferred to respective latches on the rising edge of the clock. The data is transferred to a latch according to the control bit CNT value.
3. Data [Serial Data]: You can perform either reference counter setup or programmable counter setup first.

◆ Reference: PLL set up program

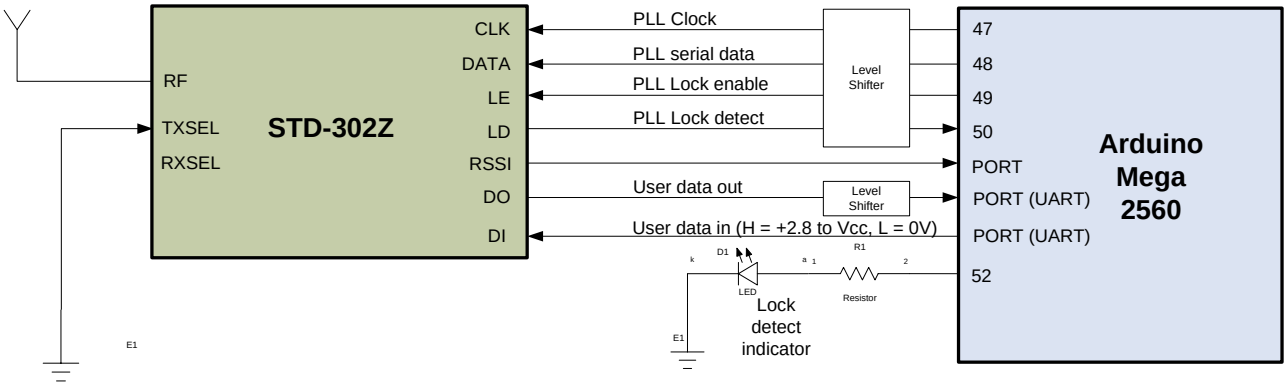
Please refer to the following PLL setting program written for Arduino MEGA 2560. Please note that the Arduino MEGA 2560 uses +5V as the high signal while the STD-302Z uses +2.8V. Therefore user needs to place a level shifter between the Arduino and the STD-302Z.
For actual use, it is necessary to observe the waveform with an oscilloscope and adjust the program to meet the conditions of the serial communication timing.

C++ header / source files are as follows. See appendix section and page no. for full code.

Header / source file	Description	Page
Board.h	Define constants and pin definitions	P10
STD_302.h / STD_302.cpp	Allows the STD-302Z to communicate with the Arduino.	P11/12
PLL.h / PLL.cpp	Contains the PLL data for all bands and channels.	P14/15
CD_Systype.h	Define system constants	P26
STD-302Z_PLL_sample.ino	The Arduino main file	P27

1. First we define our board **header.h** file so we know how to connect the module to the Arduino. By referencing the connections by name rather than pin numbers, it will become easier to understand the code. An external LED can be used to visually indicate that the PLL was successfully set.

We can now draw our connection diagram. To enable STD-302Z TX mode we can connect TXSEL to GND directly, but both TXSEL and RXSEL can be controlled by a spare port on the Arduino if desired.



2. The STD_302 header and source files contains information on how the Arduino pins should be initialised to communicate with the STD-302Z.

Function	Summary
`VOID vSTD_302_Initialize();`	Initialises the Arduino pins for communiation to STD-302Z
`VOID vSTD_302_CLKSet(INT iState);`	Outputs Hi and Lo level for the CLK signal
`VOID vSTD_302_PLLDataSet(INT iState);`	Outputs Hi and Lo level for the PLL serial data
`VOID vSTD_302_LockEnableSet(INT iState);`	Outputs the Hi signal for the Lock Enable signal
`VOID vSTD_302_RXSelect(INT iState);`	Put module into receive mode (RXSEL)
`VOID vSTD_302_TXSelect(INT iState);`	Put module into transmit mode (TXSEL)
`INT iSTD_302_LockStatus();`	Reads pin50 so Arduino can determine the lock signal. Then send command to make LED come on.

3. The PLL header and source files contain the PLL data for several bands and their channels. As the total data is quite large, this data cannot be allocated to the dynamic memory and therefore it is placed in the Arduino's static memory by using `#include <avr/pgmspace.h>`
This is no problem as this data is only the lookup data and does not change.

Function	Summary

`typedef enum{`
`E_FREQ_SEL_STD302_220 = 0,`
`E_FREQ_SEL_STD302_335,`
`E_FREQ_SEL_STD302_419,`
`E_FREQ_SEL_STD302_429,`
`E_FREQ_SEL_STD302_434,`
`E_FREQ_SEL_STD302_447,`
`E_FREQ_SEL_STD302_458,`
`E_FREQ_SEL_STD302_480,`
`E_FREQ_SEL_STD302_869,`
`E_FREQ_SEL_STD302_1216,`
`E_FREQ_SEL_STD302_1252,`
`E_FREQ_SEL_LMD400_332,`
`E_FREQ_SEL_LMD400_333,`
`E_FREQ_SEL_LMD400_334,`
`E_FREQ_SEL_LMD400_335,`
`E_FREQ_SEL_LMD400_438,`
`E_FREQ_SEL_LMD400_439,`
`E_FREQ_SEL_LMD400_440,`
`E_FREQ_SEL_LMD400_441,`
`E_FREQ_SEL_LMD400_458,`
`E_FREQ_SEL_LMD400_459,`
`E_FREQ_SEL_LMD400_460,`
`E_FREQ_SEL_LMD400_461, //22`
 	
`E_FREQ_SEL_MAX`
`} E_FREQ_SEL;`
	Allows the user to select the band depending on the version of the STD-302Z being used. In this example the 434 MHz STD-302Z is used so user can select "E_FREQ_SEL_STD302_434"

`UI_8 ucPll_PllSet(E_FREQ_SEL,UI_8);`	Select the band and channel E_FREQ_SEL : eg. E_FREQ_SEL_STD302_434 UI_8 : Channel (Unsigned 8 bit int)

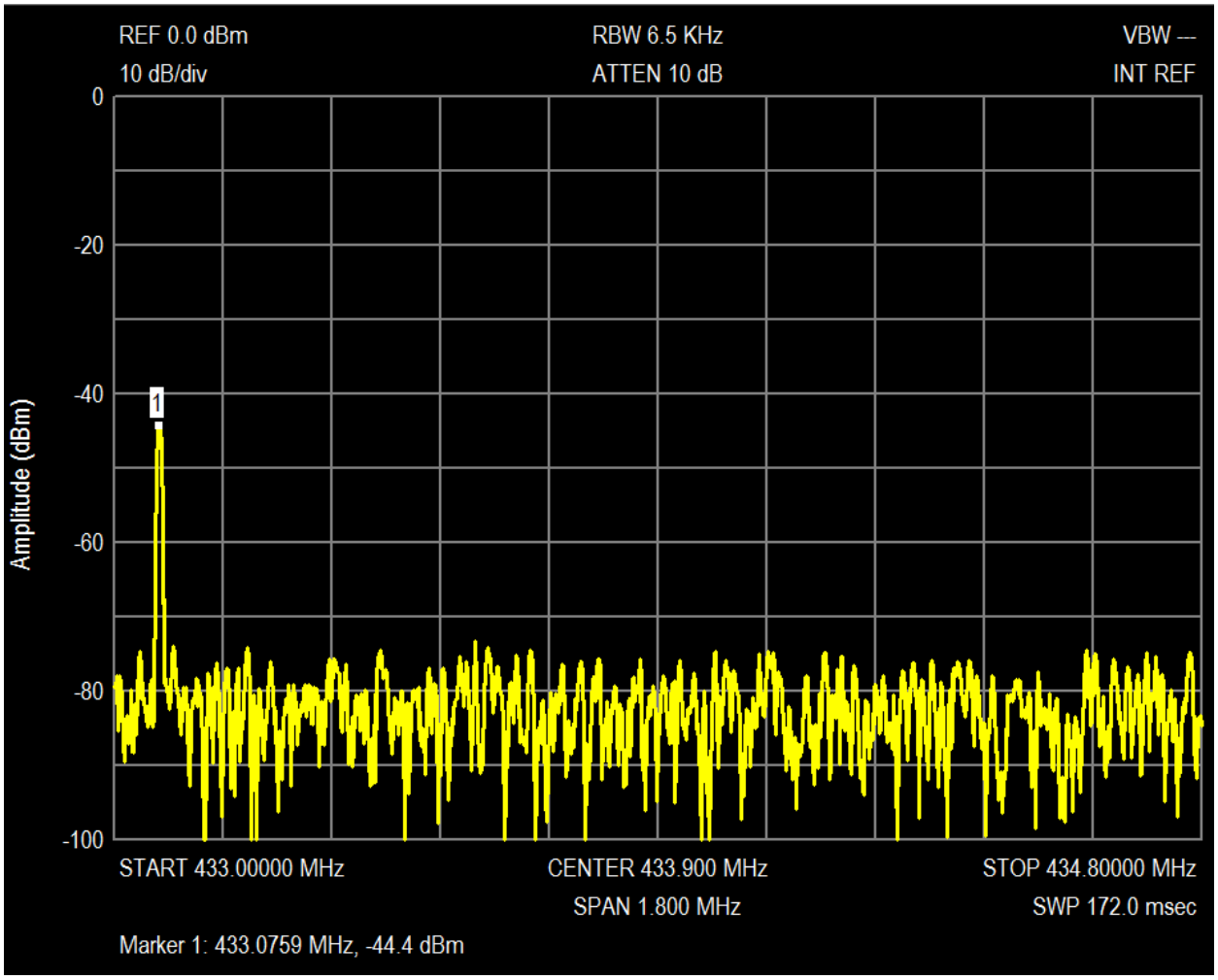
4. The file `STD-302Z_PLL_sample.ino` contains the main code and setup.

Function	Summary

`VOID setup()`	`static VOID vMain_Initialize();`
	`vSTD_302_Initialize();`
	Sets the ports as outputs and inputs
	`ucPll_PllSet(E_FREQ_SEL_STD302_434,0x00)`
	Sets the module to 434 MHz band, channel 0
	`pinMode(LED_OUTPUT,OUTPUT)`
	Sets the port for LED to indicate Lock

`VOID loop()`	This section controls transmission and reception of user data.

5. Confirmation of PLL lock on spectrum analyser: 433.075 MHz



Appendix

```
/*********************************************************************/
/*
*   @Copyright  CIRCUIT DESIGN.INC. 2018 All rights reserved.
*
*   @file      $Workfile: Board.h $
*
*   @brief     Header to define list of constants.
*
*   @author    CIRCUIT DESIGN,INC.
*
*   @note      STD-302 PLL setting
*/
/*********************************************************************/
#ifndef _BOARD_H_
#define _BOARD_H_

#include "CD_Systype.h"

/*=====================================================================
 *  #define for Global
 *=====================================================================*/

#define CLK                     (47)
#define PLL_DATA                 (48)
#define LOCK_ENABLE              (49)
#define LOCK_DETECT              (50)
#define RX_SELECT                (51)
#define LED_OUTPUT               (52)
#define TX_SELECT                (53)

#define TRUE                (1)
#define FALSE               (0)

#define PORT_LOW             (0)
#define PORT_HI              (1)


#endif // _BOARD_H_
```

```
/*****************************************************************************/
/*
*   @Copyright  CIRCUIT DESIGN.INC. 2018 All rights reserved.
*
*   @file      $Workfile: STD_302.h $
*
*   @brief     To define STD_302 connection
*
*   @author    CIRCUIT DESIGN,INC.
*
*   @note
*
*   @version   $Revision: 01$
*/
/*****************************************************************************/
#ifndef _STD_302_H_
#define _STD_302_H_

/*============================================================================
* Include File
*============================================================================*/
#include "CD_Systype.h"


/*============================================================================
* #define for Public
*============================================================================*/

/*============================================================================
* Function for Public
*============================================================================*/
VOID vSTD_302_Initialize();
VOID vSTD_302_CLKSet(INT iState);
VOID vSTD_302_PLLDataSet(INT iState);
VOID vSTD_302_LockEnableSet(INT iState);
VOID vSTD_302_RXSelect(INT iState);
VOID vSTD_302_TXSelect(INT iState);
INT iSTD_302_LockStatus();

#endif // _STD_302_H_
```

```
/*********************************************************************/
/*
*   @Copyright  CIRCUIT DESIGN.INC. 2018 All rights reserved.
*   @file      $Workfile: STD_302.cpp $
*   @brief     To define STD_302 connection.
*   @author    CIRCUIT DESIGN,INC.
*   @note
*   @version   $Revision: 01$
*********************************************************************/

/*====================================================================
* Include File
*====================================================================*/
#include "CD_Systype.h"
#include "STD_302.h"
#include "Board.h"

/*--------------------------------------------------------------------------
    @brief  Initialise.
    @param  STD-302 pins
    @retval VOID
    @note   Initialise
--------------------------------------------------------------------------*/
VOID vSTD_302_Initialize()
{
    pinMode (CLK, OUTPUT);
    pinMode (PLL_DATA, OUTPUT);
    pinMode (LOCK_ENABLE, OUTPUT);
    pinMode (LOCK_DETECT, INPUT);
    pinMode (RX_SELECT, OUTPUT);
}

/*--------------------------------------------------------------------------
    @brief  Clock Output.
    @param  iState
    @retval VOID
    @note
--------------------------------------------------------------------------*/
VOID vSTD_302_CLKSet(INT iState)
{
    digitalWrite (CLK, iState);
}

/*--------------------------------------------------------------------------
    @brief  PLL Data.
    @param  iState
    @retval VOID
    @note
--------------------------------------------------------------------------*/
VOID vSTD_302_PLLDataSet(INT iState)
{
    digitalWrite (PLL_DATA, iState);
}
```

```
/*-----------------------------------------------------------------------------
    @brief  Lock Enable.
    @param  iState
    @retval VOID
    @note
-----------------------------------------------------------------------------*/
VOID vSTD_302_LockEnableSet(INT iState)
{
    digitalWrite (LOCK_ENABLE, iState);
}

/*-----------------------------------------------------------------------------
    @brief  RX select.
    @param  iState
    @retval VOID
    @note
-----------------------------------------------------------------------------*/
VOID vSTD_302_RXSelect(INT iState)
{
    digitalWrite (RX_SELECT, iState);
}

/*-----------------------------------------------------------------------------
    @brief  TX select.
    @param  iState
    @retval VOID
    @note
-----------------------------------------------------------------------------*/
VOID vSTD_302_TXSelect(INT iState)
{
    digitalWrite (TX_SELECT, iState);
}

/*-----------------------------------------------------------------------------
    @brief  Lock Detect input.
    @param  LOCK_DETECT_INPUT
    @retval VOID
    @note
-----------------------------------------------------------------------------*/
INT iSTD_302_LockStatus()
{
    return digitalRead(LOCK DETECT);
}
```

```
/*********************************************************************/
/*
*   @Copyright  CIRCUIT DESIGN.INC. 2018 All rights reserved.
*   @file       $Workfile: Pll.h $
*   @brief      PLL Setting
*   @author     Circuit Design
*   @version    $Revision: 3 $
*   @date       $Date:: 2015-12-04 15:45:16 +0900#$
*   @note
*/
/*********************************************************************/
#ifndef    _PLL_H_
#define    _PLL_H_

/*===================================================================================
 *  Include File
 *==================================================================================*/
#include "CD_Systype.h"
/*===================================================================================
 *  #define for Public
 *==================================================================================*/
typedef enum{
   E_FREQ_SEL_STD302_220 = 0,
   E_FREQ_SEL_STD302_335,
   E_FREQ_SEL_STD302_419,
   E_FREQ_SEL_STD302_429,
   E_FREQ_SEL_STD302_434,
   E_FREQ_SEL_STD302_447,
   E_FREQ_SEL_STD302_458,
   E_FREQ_SEL_STD302_480,
   E_FREQ_SEL_STD302_869,
   E_FREQ_SEL_STD302_1216,
   E_FREQ_SEL_STD302_1252,
   E_FREQ_SEL_LMD400_332,
   E_FREQ_SEL_LMD400_333,
   E_FREQ_SEL_LMD400_334,
   E_FREQ_SEL_LMD400_335,
   E_FREQ_SEL_LMD400_438,
   E_FREQ_SEL_LMD400_439,
   E_FREQ_SEL_LMD400_440,
   E_FREQ_SEL_LMD400_441,
   E_FREQ_SEL_LMD400_458,
   E_FREQ_SEL_LMD400_459,
   E_FREQ_SEL_LMD400_460,
   E_FREQ_SEL_LMD400_461, //22

   E_FREQ_SEL_MAX
} E_FREQ_SEL;
/*===================================================================================
 *  Function for Public
 *==================================================================================*/
UI_8 ucPll_PllSet(E_FREQ_SEL,UI_8);
#endif     /* _PLL_H_ */
```

```
/*********************************************************************/
/*
*   @Copyright  CIRCUIT DESIGN.INC. 2018 All rights reserved.
*
*   @file      $Workfile: Pll.c $
*
*   @brief     PLL Setting
*
*   @author    Circuit Design
*
*   @version   $Revision: 3 $
*
*   @date      $Date:: 2015-12-04 15:45:16 +0900#$
*
*   @note
*/
/*********************************************************************/
/*====================================================================
* Include File
*====================================================================*/

#include "Board.h"
#include "Pll.h"
#include "STD_302.h"
#include <avr/pgmspace.h>



/*====================================================================
* #define for Local
*====================================================================*/
#define REF_MIN     (3)
#define REF_MAX     (16383)
#define PROG_MIN    (3)
#define PROG_MAX    (2047)
#define SWAL_MAX    (127)

#define CS         (0)
#define LDS        (0)
#define FC         (1)
#define CNT REF     (1)
#define CNT_PRG     (0)

#define DATA_SIZE   (19)

#define STD_302_SIZE_220    (6)
#define STD_302_SIZE_335    (6)
#define STD_302_SIZE_419    (29)
#define STD_302_SIZE_429    (47)
#define STD_302_SIZE_434    (69)
#define STD_302_SIZE_447    (58)
#define STD_302_SIZE_458    (27)
#define STD_302_SIZE_480    (64)
#define STD_302_SIZE_869    (79)
#define STD_302_SIZE_1216   (41)
#define STD_302_SIZE_1252   (41)
```

```
#define LMD_400_SIZE_332    (100)
#define LMD_400_SIZE_333    (100)
#define LMD_400_SIZE_334    (100)
#define LMD_400_SIZE_335    (100)
#define LMD_400_SIZE_438    (100)
#define LMD_400_SIZE_439    (100)
#define LMD_400_SIZE_440    (100)
#define LMD_400_SIZE_441    (21)
#define LMD_400_SIZE_458    (100)
#define LMD_400_SIZE_459    (100)
#define LMD_400_SIZE_460    (100)
#define LMD_400_SIZE_461    (61)

typedef struct{
    UI_16 unProgramableCount;
    UI_8  ucSwallowCount;
} T_COUNT_DATA;

typedef struct{
    T_COUNT_DATA sSTD_302_220[STD_302_SIZE_220];
    T_COUNT_DATA sSTD_302_335[STD_302_SIZE_335];
    T_COUNT_DATA sSTD_302_419[STD_302_SIZE_419];
    T_COUNT_DATA sSTD_302_429[STD_302_SIZE_429];
    T_COUNT_DATA sSTD_302_434[STD_302_SIZE_434];
    T_COUNT_DATA sSTD_302_447[STD_302_SIZE_447];
    T_COUNT_DATA sSTD_302_458[STD_302_SIZE_458];
    T_COUNT_DATA sSTD_302_480[STD_302_SIZE_480];
    T_COUNT_DATA sSTD_302_869[STD_302_SIZE_869];
    T_COUNT_DATA sSTD_302_1216[STD_302_SIZE_1216];
    T_COUNT_DATA sSTD_302_1252[STD_302_SIZE_1252];
    T_COUNT_DATA sLMD_400_332[LMD_400_SIZE_332];
    T_COUNT_DATA sLMD_400_333[LMD_400_SIZE_333];
    T_COUNT_DATA sLMD_400_334[LMD_400_SIZE_334];
    T_COUNT_DATA sLMD_400_335[LMD_400_SIZE_335];
    T_COUNT_DATA sLMD_400_438[LMD_400_SIZE_438];
    T_COUNT_DATA sLMD_400_439[LMD_400_SIZE_439];
    T_COUNT_DATA sLMD_400_440[LMD_400_SIZE_440];
    T_COUNT_DATA sLMD_400_441[LMD_400_SIZE_441];
    T_COUNT_DATA sLMD_400_458[LMD_400_SIZE_458];
    T_COUNT_DATA sLMD_400_459[LMD_400_SIZE_459];
    T_COUNT_DATA sLMD_400_460[LMD_400_SIZE_460];
    T COUNT DATA sLMD 400 461[LMD 400 SIZE 461];

} T_PROGRAMABLE_COUNT;

/*=========================================================================
* Global Variable
*=========================================================================*/

/*=========================================================================
* Static Variable
*=========================================================================*/
static UI_8 st_ucData[DATA_SIZE] = {0};
static const T_PROGRAMABLE_COUNT st_sPllData PROGMEM ={
    {{251,48},{252,32},{252,54},{253,40},{254,10},{254,34}},
    {{392,33},{392,35},{392,37},{392,39},{392,41},{392,43}},
```

{{248,9},{248,10},{248,11},{248,12},{248,13},{248,14},{248,15},{248,16},{248,17},{248,18},{248,19},{248,20},{248,21},{248,22},{248,23},{248,24},{248,25},{248,26},{248,27},{248,28},{248,29},{248,30},{248,31},{248,32},{248,33},{248,34},{248,35},{248,36},{248,37}},

{{509,28},{509,22},{509,23},{509,24},{509,25},{509,26},{509,27},{509,28},{509,29},{509,30},{509,31},{509,32},{509,33},{509,34},{509,35},{509,36},{509,37},{509,38},{509,39},{509,40},{509,41},{509,42},{509,43},{509,44},{509,45},{509,46},{509,47},{509,48},{509,49},{509,50},{509,51},{509,52},{509,53},{509,54},{509,55},{509,56},{509,57},{509,58},{509,59},{509,60},{509,61},{509,62},{509,63},{509,64},{510,1},{510,2},{510,3}},

{{257,7},{257,8},{257,9},{257,10},{257,11},{257,12},{257,13},{257,14},{257,15},{257,16},{257,17},{257,18},{257,19},{257,20},{257,21},{257,22},{257,23},{257,24},{257,25},{257,26},{257,27},{257,28},{257,29},{257,30},{257,31},{257,32},{257,33},{257,34},{257,35},{257,36},{257,37},{257,38},{257,39},{257,40},{257,41},{257,42},{257,43},{257,44},{257,45},{257,46},{257,47},{257,48},{257,49},{257,50},{257,51},{257,52},{257,53},{257,54},{257,55},{257,56},{257,57},{257,58},{257,59},{257,60},{257,61},{257,62},{257,63},{257,64},{258,1},{258,2},{258,3},{258,4},{258,5},{258,6},{258,7},{258,8},{258,9},{258,10},{258,11}},

{{531,62},{531,63},{532,0},{532,1},{532,2},{532,3},{532,4},{532,5},{532,6},{532,7},{532,8},{532,9},{532,10},{532,11},{532,12},{532,13},{532,14},{532,15},{532,16},{532,17},{532,18},{532,19},{532,20},{532,21},{532,22},{532,23},{532,24},{532,25},{532,26},{532,27},{532,28},{532,29},{532,30},{532,31},{532,32},{532,33},{532,34},{532,35},{532,36},{532,37},{532,38},{532,39},{532,40},{532,41},{532,42},{532,43},{532,44},{532,45},{532,46},{532,47},{532,48},{532,49},{532,50},{532,51},{532,52},{532,53},{532,54},{532,55}},

{{273,1},{273,2},{273,3},{273,4},{273,5},{273,6},{273,7},{273,8},{273,9},{273,10},{273,11},{273,12},{273,13},{273,14},{273,15},{273,16},{273,17},{273,18},{273,19},{273,20},{273,21},{273,22},{273,23},{273,24},{273,25},{273,26},{273,27}},

{{572,56},{572,57},{572,58},{572,59},{572,60},{572,61},{572,62},{572,63},{573,0},{573,1},{573,2},{573,3},{573,4},{573,5},{573,6},{573,7},{573,8},{573,9},{573,10},{573,11},{573,12},{573,13},{573,14},{573,15},{573,16},{573,17},{573,18},{573,19},{573,20},{573,21},{573,22},{573,23},{573,24},{573,25},{573,26},{573,27},{573,28},{573,29},{573,30},{573,31},{573,32},{573,33},{573,34},{573,35},{573,36},{573,37},{573,38},{573,39},{573,40},{573,41},{573,42},{573,43},{573,44},{573,45},{573,46},{573,47},{573,48},{573,49},{573,50},{573,51},{573,52},{573,53},{573,54},{573,55}},

{{528,61},{528,62},{528,63},{529,0},{529,1},{529,2},{529,3},{529,4},{529,5},{529,6},{529,7},{529,8},{529,9},{529,10},{529,11},{529,12},{529,13},{529,14},{529,15},{529,16},{529,17},{529,18},{529,19},{529,20},{529,21},{529,22},{529,23},{529,24},{529,25},{529,26},{529,27},{529,28},{529,29},{529,30},{529,31},{529,32},{529,33},{529,34},{529,35},{529,36},{529,37},{529,38},{529,39},{529,40},{529,41},{529,42},{529,43},{529,44},{529,45},{529,46},{529,47},{529,48},{529,49},{529,50},{529,51},{529,52},{529,53},{529,54},{529,55},{529,56},{529,57},{529,58},{529,59},{529,60},{529,61},{529,62},{529,63},{529,64},{530,1},{530,2},{530,3},{530,4},{530,5},{530,6},{530,7},{530,8},{530,9},{530,10},{530,11}},

{{1492,57},{1492,57},{1492,59},{1492,61},{1492,63},{1493,1},{1493,3},{1493,5},{1493,7},{1493,9},{1493,11},{1493,13},{1493,15},{1493,17},{1493,19},{1493,21},{1493,23},{1493,25},{1493,27},{1493,29},{1493,31},{1493,33},{1493,35},{1493,37},{1493,39},{1493,41},{1493,43},{1493,45},{1493,47},{1493,49},{1493,51},{1493,53},{1493,55},{1493,57},{1493,59},{1493,61},{1493,63},{1494,1},{1494,3},{1494,5},{1494,7}},

{{1537,57},{1537,57},{1537,59},{1537,61},{1537,63},{1538,1},{1538,3},{1538,5},{1538,7},{1538,9},{1538,11},{1538,13},{1538,15},{1538,17},{1538,19},{1538,21},{1538,23},{1538,25},{1538,27},{1538,29},{1538,31},{1538,33},{1538,35},{1538,37},{1538,39},{1538,41},{1538,43},{1538,45},{1538,47},{1538,49},{1538,51},{1538,53},{1538,55},{1538,57},{1538,59},{1538,61},{1538,63},{1539,1},{1539,3},{1539,5},{1539,7}},

{{387,56},{387,57},{387,58},{387,59},{387,60},{387,61},{387,62},{387,63},{388,0},{388,1},{388,2},{388,3},{388,4},{388,5},{388,6},{388,7},{388,8},{388,9},{388,10},{388,11},{388,12},{388,13},{388,14},{388,15},{388,16},{388,17},{388,18},{388,19},{388,20},{388,21},{388,22},{388,23},{388,24},{388,25},{388,26},{388,27},{388,28},{388,29},{388,30},{388,31},{388,32},{388,33},{388,34},{388,35},{388,36},{388,37},{388,38},{388,39},{388,40},{388,41},{388,42},{388,43},{388,44},{388,45},{388,46},{388,47},{388,48},{388,49},{388,50},{388,51},{388,52},{388,53},{388,54},{388,55},{388,56},{388,57},{388,58},{388,59},{388,60},{388,61},{388,62},{388,63},{388,64},{389,1},{389,2},{389,3},{389,4},{389,5},{389,6},{389,7},{389,8},{389,9},{389,10},{389,11},{389,12},{389,13},{389,14},{389,15},{389,16},{389,17},{389,18},{389,19},{389,20},{389,21},{389,22},{389,23},{389,24},{389,25},{389,26},{389,27}},

{{389,28},{389,29},{389,30},{389,31},{389,32},{389,33},{389,34},{389,35},{389,36},{389,37},{389,38},{389,39},{389,40},{389,41},{389,42},{389,43},{389,44},{389,45},{389,46},{389,47},{389,48},{389,49},{389,50},{389,51},{389,52},{389,53},{389,54},{389,55},{389,56},{389,57},{389,58},{389,59},{389,60},{389,61},{389,62},{389,63},{389,64},{390,1},{390,2},{390,3},{390,4},{390,5},{390,6},{390,7},{390,8},{390,9},{390,10},{390,11},{390,12},{390,13},{390,14},{390,15},{390,16},{390,17},{390,18},{390,19},{390,20},{390,21},{390,22},{390,23},{390,24},{390,25},{390,26},{390,27},{390,28},{390,29},{390,30},{390,31},{390,32},{390,33},{390,34},{390,35},{390,36},{390,37},{390,38},{390,39},{390,40},{390,41},{390,42},{390,43},{390,44},{390,45},{390,46},{390,47},{390,48},{390,49},{390,50},{390,51},{390,52},{390,53},{390,54},{390,55},{390,56},{390,57},{390,58},{390,59},{390,60},{390,61},{390,62},{390,63}},

{{391,0},{391,1},{391,2},{391,3},{391,4},{391,5},{391,6},{391,7},{391,8},{391,9},{391,10},{391,11},{391,12},{391,13},{391,14},{391,15},{391,16},{391,17},{391,18},{391,19},{391,20},{391,21},{391,22},{391,23},{391,24},{391,25},{391,26},{391,27},{391,28},{391,29},{391,30},{391,31},{391,32},{391,33},{391,34},{391,35},{391,36},{391,37},{391,38},{391,39},{391,40},{391,41},{391,42},{391,43},{391,44},{391,45},{391,46},{391,47},{391,48},{391,49},{391,50},{391,51},{391,52},{391,53},{391,54},{391,55},{391,56},{391,57},{391,58},{391,59},{391,60},{391,61},{391,62},{391,63},{391,64},{392,1},{392,2},{392,3},{392,4},{392,5},{392,6},{392,7},{392,8},{392,9},{392,10},{392,11},{392,12},{392,13},{392,14},{392,15},{392,16},{392,17},{392,18},{392,19},{392,20},{392,21},{392,22},{392,23},{392,24},{392,25},{392,26},{392,27},{392,28},{392,29},{392,30},{392,31},{392,32},{392,33},{392,34},{392,35},},

{{392,36},{392,37},{392,38},{392,39},{392,40},{392,41},{392,42},{392,43},{392,44},{392,45},{392,46},{392,47},{392,48},{392,49},{392,50},{392,51},{392,52},{392,53},{392,54},{392,55},{392,56},{392,57},{392,58},{392,59},{392,60},{392,61},{392,62},{392,63},{393,0},{393,1},{393,2},{393,3},{393,4},{393,5},{393,6},{393,7},{393,8},{393,9},{393,10},{393,11},{393,12},{393,13},{393,14},{393,15},{393,16},{393,17},{393,18},{393,19},{393,20},{393,21},{393,22},{393,23},{393,24},{393,25},{393,26},{393,27},{393,28},{393,29},{393,30},{393,31},{393,32},{393,33},{393,34},{393,35},{393,36},{393,37},{393,38},{393,39},{393,40},{393,41},{393,42},{393,43},{393,44},{393,45},{393,46},{393,47},{393,48},{393,49},{393,50},{393,51},{393,52},{393,53},{393,54},{393,55},{393,56},{393,57},{393,58},{393,59},{393,60},{393,61},{393,62},{393,63},{393,64},{394,1},{394,2},{394,3},{394,4},{394,5},{394,6},{394,7}},

{{520,24},{520,25},{520,26},{520,27},{520,28},{520,29},{520,30},{520,31},{520,32},{520,33},{520,34},{520,35},{520,36},{520,37},{520,38},{520,39},{520,40},{520,41},{520,42},{520,43},{520,44},{520,45},{520,46},{520,47},{520,48},{520,49},{520,50},{520,51},{520,52},{520,53},{520,54},{520,55},{520,56},{520,57},{520,58},{520,59},{520,60},{520,61},{520,62},{520,63},{520,64},{521,1},{521,2},{521,3},{521,4},{521,5},{521,6},{521,7},{521,8},{521,9},{521,10},{521,11},{521,12},{521,13},{521,14},{521,15},{521,16},{521,17},{521,18},{521,19},{521,20},{521,21},{521,22},{521,23},{521,24},{521,25},{521,26},{521,27},{521,28},{521,29},{521,30},{521,31},{521,32},{521,33},{521,34},{521,35},{521,36},{521,37},{521,38},{521,39},{521,40},{521,41},{521,42},{521,43},{521,44},{521,45},{521,46},{521,47},{521,48},{521,49},{521,50},{521,51},{521,52},{521,53},{521,54},{521,55},{521,56},{521,57},{521,58},{521,59}},

{{521,60},{521,61},{521,62},{521,63},{522,0},{522,1},{522,2},{522,3},{522,4},{522,5},{522,6},{522,7},{522,8},{522,9},{522,10},{522,11},{522,12},{522,13},{522,14},{522,15},{522,16},{522,17},{522,18},{522,19},{522,20},{522,21},{522,22},{522,23},{522,24},{522,25},{522,26},{522,27},{522,28},{522,29},{522,30},{522,31},{522,32},{522,33},{522,34},{522,35},{522,36},{522,37},{522,38},{522,39},{522,40},{522,41},{522,42},{522,43},{522,44},{522,45},{522,46},{522,47},{522,48},{522,49},{522,50},{522,51},{522,52},{522,53},{522,54},{522,55},{522,56},{522,57},{522,58},{522,59},{522,60},{522,61},{522,62},{522,6

```
3},{522,64},{523,1},{523,2},{523,3},{523,4},{523,5},{523,6},{523,7},{523,8},{523,9},{523,10},{523,11
},{523,12},{523,13},{523,14},{523,15},{523,16},{523,17},{523,18},{523,19},{523,20},{523,21},{523,22}
,{523,23},{523,24},{523,25},{523,26},{523,27},{523,28},{523,29},{523,30},{523,31}},

{{523,32},{523,33},{523,34},{523,35},{523,36},{523,37},{523,38},{523,39},{523,40},{523,41},{523,42},
{523,43},{523,44},{523,45},{523,46},{523,47},{523,48},{523,49},{523,50},{523,51},{523,52},{523,53},{
523,54},{523,55},{523,56},{523,57},{523,58},{523,59},{523,60},{523,61},{523,62},{523,63},{524,0},{52
4,1},{524,2},{524,3},{524,4},{524,5},{524,6},{524,7},{524,8},{524,9},{524,10},{524,11},{524,12},{524
,13},{524,14},{524,15},{524,16},{524,17},{524,18},{524,19},{524,20},{524,21},{524,22},{524,23},{524,
24},{524,25},{524,26},{524,27},{524,28},{524,29},{524,30},{524,31},{524,32},{524,33},{524,34},{524,3
5},{524,36},{524,37},{524,38},{524,39},{524,40},{524,41},{524,42},{524,43},{524,44},{524,45},{524,46
},{524,47},{524,48},{524,49},{524,50},{524,51},{524,52},{524,53},{524,54},{524,55},{524,56},{524,57}
,{524,58},{524,59},{524,60},{524,61},{524,62},{524,63},{524,64},{525,1},{525,2},{525,3}},

{{525,4},{525,5},{525,6},{525,7},{525,8},{525,9},{525,10},{525,11},{525,12},{525,13},{525,14},{525,1
5},{525,16},{525,17},{525,18},{525,19},{525,20},{525,21},{525,22},{525,23},{525,24}},

{{545,24},{545,25},{545,26},{545,27},{545,28},{545,29},{545,30},{545,31},{545,32},{545,33},{545,34},
{545,35},{545,36},{545,37},{545,38},{545,39},{545,40},{545,41},{545,42},{545,43},{545,44},{545,45},{
545,46},{545,47},{545,48},{545,49},{545,50},{545,51},{545,52},{545,53},{545,54},{545,55},{545,56},{5
45,57},{545,58},{545,59},{545,60},{545,61},{545,62},{545,63},{545,64},{546,1},{546,2},{546,3},{546,4
},{546,5},{546,6},{546,7},{546,8},{546,9},{546,10},{546,11},{546,12},{546,13},{546,14},{546,15},{546
,16},{546,17},{546,18},{546,19},{546,20},{546,21},{546,22},{546,23},{546,24},{546,25},{546,26},{546,
27},{546,28},{546,29},{546,30},{546,31},{546,32},{546,33},{546,34},{546,35},{546,36},{546,37},{546,3
8},{546,39},{546,40},{546,41},{546,42},{546,43},{546,44},{546,45},{546,46},{546,47},{546,48},{546,49
},{546,50},{546,51},{546,52},{546,53},{546,54},{546,55},{546,56},{546,57},{546,58},{546,59}},

{{546,60},{546,61},{546,62},{546,63},{547,0},{547,1},{547,2},{547,3},{547,4},{547,5},{547,6},{547,7}
,{547,8},{547,9},{547,10},{547,11},{547,12},{547,13},{547,14},{547,15},{547,16},{547,17},{547,18},{5
47,19},{547,20},{547,21},{547,22},{547,23},{547,24},{547,25},{547,26},{547,27},{547,28},{547,29},{54
7,30},{547,31},{547,32},{547,33},{547,34},{547,35},{547,36},{547,37},{547,38},{547,39},{547,40},{547
,41},{547,42},{547,43},{547,44},{547,45},{547,46},{547,47},{547,48},{547,49},{547,50},{547,51},{547,
52},{547,53},{547,54},{547,55},{547,56},{547,57},{547,58},{547,59},{547,60},{547,61},{547,62},{547,6
3},{547,64},{548,1},{548,2},{548,3},{548,4},{548,5},{548,6},{548,7},{548,8},{548,9},{548,10},{548,11
},{548,12},{548,13},{548,14},{548,15},{548,16},{548,17},{548,18},{548,19},{548,20},{548,21},{548,22}
,{548,23},{548,24},{548,25},{548,26},{548,27},{548,28},{548,29},{548,30},{548,31}},

{{548,32},{548,33},{548,34},{548,35},{548,36},{548,37},{548,38},{548,39},{548,40},{548,41},{548,42},
{548,43},{548,44},{548,45},{548,46},{548,47},{548,48},{548,49},{548,50},{548,51},{548,52},{548,53},{
548,54},{548,55},{548,56},{548,57},{548,58},{548,59},{548,60},{548,61},{548,62},{548,63},{549,0},{54
9,1},{549,2},{549,3},{549,4},{549,5},{549,6},{549,7},{549,8},{549,9},{549,10},{549,11},{549,12},{549
,13},{549,14},{549,15},{549,16},{549,17},{549,18},{549,19},{549,20},{549,21},{549,22},{549,23},{549,
24},{549,25},{549,26},{549,27},{549,28},{549,29},{549,30},{549,31},{549,32},{549,33},{549,34},{549,3
5},{549,36},{549,37},{549,38},{549,39},{549,40},{549,41},{549,42},{549,43},{549,44},{549,45},{549,46
},{549,47},{549,48},{549,49},{549,50},{549,51},{549,52},{549,53},{549,54},{549,55},{549,56},{549,57}
,{549,58},{549,59},{549,60},{549,61},{549,62},{549,63},{549,64},{550,1},{550,2},{550,3}},

{{550,4},{550,5},{550,6},{550,7},{550,8},{550,9},{550,10},{550,11},{550,12},{550,13},{550,14},{550,1
5},{550,16},{550,17},{550,18},{550,19},{550,20},{550,21},{550,22},{550,23},{550,24},{550,25},{550,26
},{550,27},{550,28},{550,29},{550,30},{550,31},{550,32},{550,33},{550,34},{550,35},{550,36},{550,37}
,{550,38},{550,39},{550,40},{550,41},{550,42},{550,43},{550,44},{550,45},{550,46},{550,47},{550,48},
{550,49},{550,50},{550,51},{550,52},{550,53},{550,54},{550,55},{550,56},{550,57},{550,58},{550,59},{
550,60},{550,61},{550,62},{550,63},{550,64}}
};
```

```
static const UI_16 st_unRefCnt[E_FREQ_SEL_MAX] PROGMEM =
{1700,1700,850,1700,850,1700,850,1700,850,1700,1700,1700,1700,1700,1700,1700,1700,1700,1700,170
0,1700,1700};


/*===========================================================================
* Prototype Declaration
*===========================================================================*/
static VOID vPll_SetRefData(UI_16,UI_8);
static VOID vPll_SetProgData(UI_16,UI_8);

/*---------------------------------------------------------------------------
    @brief  Reference counter setting generator
    @param  unRef   Reference Counter
    @param  unPre   Prescaler
    @return none
    @note

---------------------------------------------------------------------------*/
static VOID vPll_SetRefData(UI_16 unRef,UI_8 ucPre)
{
    UI_8 ucCnt = 0;
    UI_16 unRefBuf = 0;

    st_ucData[0] = CNT_REF;
    if(0 == ucPre){
        st_ucData[15] = 0;
    }else{
        st_ucData[15] = 1;
    }
    st_ucData[16] = FC;
    st_ucData[17] = LDS;
    st_ucData[18] = CS;

    unRefBuf = unRef;
    for(ucCnt = 1;ucCnt < 15;ucCnt++){  //Generate serial data
        if(unRefBuf <= 0){
            st_ucData[ucCnt] = 0;
        } else {
            st_ucData[ucCnt] = (UI_8)(unRefBuf%2);
            unRefBuf /= 2;
        }
    }

}

/*---------------------------------------------------------------------------
    @brief  Programmable Counter Setting Data Generation
    @param  unProg  Programmable Counter
    @param  ucSwal  Swallow Counter
    @return None
    @note
---------------------------------------------------------------------------*/
static VOID vPll_SetProgData(UI_16 unProg,UI_8 ucSwal)
{
    UI_8 ucCnt = 0;        //Loop Counter
    UI_8 ucSwalBuf = 0;    //Swallow Counter Buffer
```

```
    UI_16 unProgBuf = 0;    //Programmable Counter Buffer

    st_ucData[0] = CNT_PRG;

    ucSwalBuf = ucSwal;
    unProgBuf = unProg;

    for(ucCnt = 1;ucCnt < 8;ucCnt++){
        if(ucSwalBuf <= 0){
            st_ucData[ucCnt] = 0;
        } else {
            st_ucData[ucCnt] = ucSwalBuf%2;
            ucSwalBuf /= 2;
        }
    }

    for(ucCnt = 8;ucCnt < 19;ucCnt++){
        if(unProgBuf <= 0){
            st_ucData[ucCnt] = 0;
        } else {
            st_ucData[ucCnt] = (UI_8)(unProgBuf%2);
            unProgBuf /= 2;
        }
    }
}

/*-------------------------------------------------------------------------------
    @brief  PLL Setting
    @param  eFreqSel    Freq Band Select
    @param  ucCh    Channel
    @retval TRUE    Success
    @retval FALSE   Fail
    @note
-------------------------------------------------------------------------------*/
UI_8 ucPll_PllSet(E_FREQ_SEL eFreqSel,UI_8 ucCh)
{
    UI_8 ucCnt = 0;
    UI_16 unRef;
    UI_8 ucPre = 64;
    UI_16 unProg;
    UI_8 ucSwal;
    switch(eFreqSel){
        case E_FREQ_SEL_STD302_220:

            unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
            if(STD_302_SIZE_220 <= ucCh){
                ucCh = 0;
            }


            unProg = pgm_read_word(&(st_sPllData.sSTD_302_220[ucCh].unProgramableCount));
            ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_220[ucCh].ucSwallowCount));
            break;
        case E_FREQ_SEL_STD302_335:

            unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
            if(STD_302_SIZE_335 <= ucCh){
```

```
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_335[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_335[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_STD302_419:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(STD_302_SIZE_419 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_419[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_419[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_STD302_429:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(STD_302_SIZE_429 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_429[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_429[ucCh].ucSwallowCount));
        break;



    case E_FREQ_SEL_STD302_434:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(STD_302_SIZE_434 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_434[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_434[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_STD302_447:

        unRef = pgm read word(&st unRefCnt[eFreqSel]);
        if(STD_302_SIZE_447 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_447[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_447[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_STD302_458:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(STD_302_SIZE_458 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_458[ucCh].unProgramableCount));
```

```
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_458[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_STD302_480:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(STD_302_SIZE_480 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_480[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_480[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_STD302_869:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(STD_302_SIZE_869 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_869[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_869[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_STD302_1216:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(STD_302_SIZE_1216 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_1216[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_1216[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_STD302_1252:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(STD_302_SIZE_1252 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sSTD_302_1252[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sSTD_302_1252[ucCh].ucSwallowCount));
        break;

    case E_FREQ_SEL_LMD400_438:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(LMD_400_SIZE_438 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sLMD_400_438[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sLMD_400_438[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_LMD400_439:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(LMD_400_SIZE_439 <= ucCh){
```

```
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sLMD_400_439[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sLMD_400_439[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_LMD400_440:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(LMD_400_SIZE_440 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sLMD_400_440[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sLMD_400_440[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_LMD400_441:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(LMD_400_SIZE_441 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sLMD_400_441[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sLMD_400_441[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_LMD400_458:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(LMD_400_SIZE_458 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sLMD_400_458[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sLMD_400_458[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_LMD400_459:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(LMD_400_SIZE_459 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sLMD_400_459[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sLMD_400_459[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_LMD400_460:

        unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
        if(LMD_400_SIZE_460 <= ucCh){
            ucCh = 0;
        }

        unProg = pgm_read_word(&(st_sPllData.sLMD_400_460[ucCh].unProgramableCount));
        ucSwal = pgm_read_byte(&(st_sPllData.sLMD_400_460[ucCh].ucSwallowCount));
        break;
    case E_FREQ_SEL_LMD400_461:
```

```
            unRef = pgm_read_word(&st_unRefCnt[eFreqSel]);
            if(LMD_400_SIZE_461 <= ucCh){
                ucCh = 0;
            }

            unProg = pgm_read_word(&(st_sPllData.sLMD_400_461[ucCh].unProgramableCount));
            ucSwal = pgm_read_byte(&(st_sPllData.sLMD_400_461[ucCh].ucSwallowCount));
            break;
        default:
            return FALSE;
            break;
    }

    vSTD_302_LockEnableSet(PORT_LOW);
    vSTD_302_CLKSet(PORT_LOW);
    vPll_SetRefData(unRef,ucPre);


    for(ucCnt = 1;ucCnt <= DATA_SIZE;ucCnt++){

        vSTD_302_PLLDataSet(st_ucData[DATA_SIZE - ucCnt]);
        vSTD_302_CLKSet(PORT_HI);
        vSTD_302_CLKSet(PORT_LOW);

    }

    vSTD_302_LockEnableSet(PORT_HI);
    vSTD_302_LockEnableSet(PORT_LOW);

    vPll_SetProgData(unProg,ucSwal);

    for(ucCnt = 1;ucCnt <= DATA_SIZE;ucCnt++){

        vSTD_302_PLLDataSet(st_ucData[DATA_SIZE - ucCnt]);
        vSTD_302_CLKSet(PORT_HI);
        vSTD_302_CLKSet(PORT_LOW);

    }
    vSTD_302_LockEnableSet(PORT_HI);
    vSTD_302_LockEnableSet(PORT_LOW);

    if(iSTD_302_LockStatus() == PORT_HI){
        return TRUE;
    }else{
        return FALSE;
    }
}
```

```
/*********************************************************************/
/*
*   @Copyright  CIRCUIT DESIGN.INC. 2014 All rights reserved.
*
*   @file      $Workfile: CD_Systype.h $
*
*   @brief     Header to define list of definition types.
*
*   @author    CURCUIT DESIGN,INC.
*
*   @note
*/
/*********************************************************************/

#ifndef _CD_SYSTYPE_H_
#define _CD_SYSTYPE_H_

/*====================================================================
 *  Include File
 *====================================================================*/
#include <arduino.h>

/*====================================================================================
 *  Definition for Package
 *====================================================================================*/
typedef void                VOID;
typedef signed int           INT;
typedef unsigned int         UINT;
typedef signed char          CHAR;
typedef signed char          SI_8;
typedef unsigned char        UCHAR;
typedef unsigned char        UI_8;
typedef signed short         SHORT;
typedef signed short         SI_16;
typedef unsigned short       USHORT;
typedef unsigned short       UI_16;
typedef signed long          LONG;
typedef signed long          SI_32;
typedef unsigned long        ULONG;
typedef unsigned long        UI_32;
typedef double              DOUBLE;
typedef float               FLOAT;
typedef boolean              BOOL;
typedef String              STRING;

#endif // _CD_SYSTYPE_H_
```

```
/*********************************************************************/
/*
*   @Copyright  CIRCUIT DESIGN.INC. 2018 All rights reserved.
*
*   @file      $Workfile: STD-302Z_PLL_sample.ino $
*
*   @brief     Main program.
*
*   @author    CIRCUIT DESIGN,INC.
*
*   @note
*/
/*********************************************************************/


/*====================================================================
 *  Include File
 *====================================================================*/

 #include "CD_Systype.h"
 #include "Board.h"
 #include "Pll.h"
 #include "STD_302.h"
 #include <avr/pgmspace.h>

static VOID vMain_Initialize();

VOID setup()
{
    vMain_Initialize();
    while(!iSTD_302_LockStatus()){} //wait for PLL lock
    digitalWrite(LED_OUTPUT,TRUE); //use LED to indicate PLL lock
}

VOID loop()
{
    //Send and receive user data goes here
}

/*-------------------------------------------------------------------------------
    @brief  Main initialize

    @param  VOID

    @retval VOID

    @note
-------------------------------------------------------------------------------*/
static VOID vMain_Initialize()
{
    vSTD_302_Initialize();
    ucPll_PllSet(E_FREQ_SEL_STD302_434,0x00); //Set to 434 MHz band, 433.075 MHz (Ch 0x00)
    pinMode(LED_OUTPUT,OUTPUT); //initialise terminal for LED
}
```