

美作アマチュア無線クラブ(JA4YAM) 第7回デジタル技術勉強会

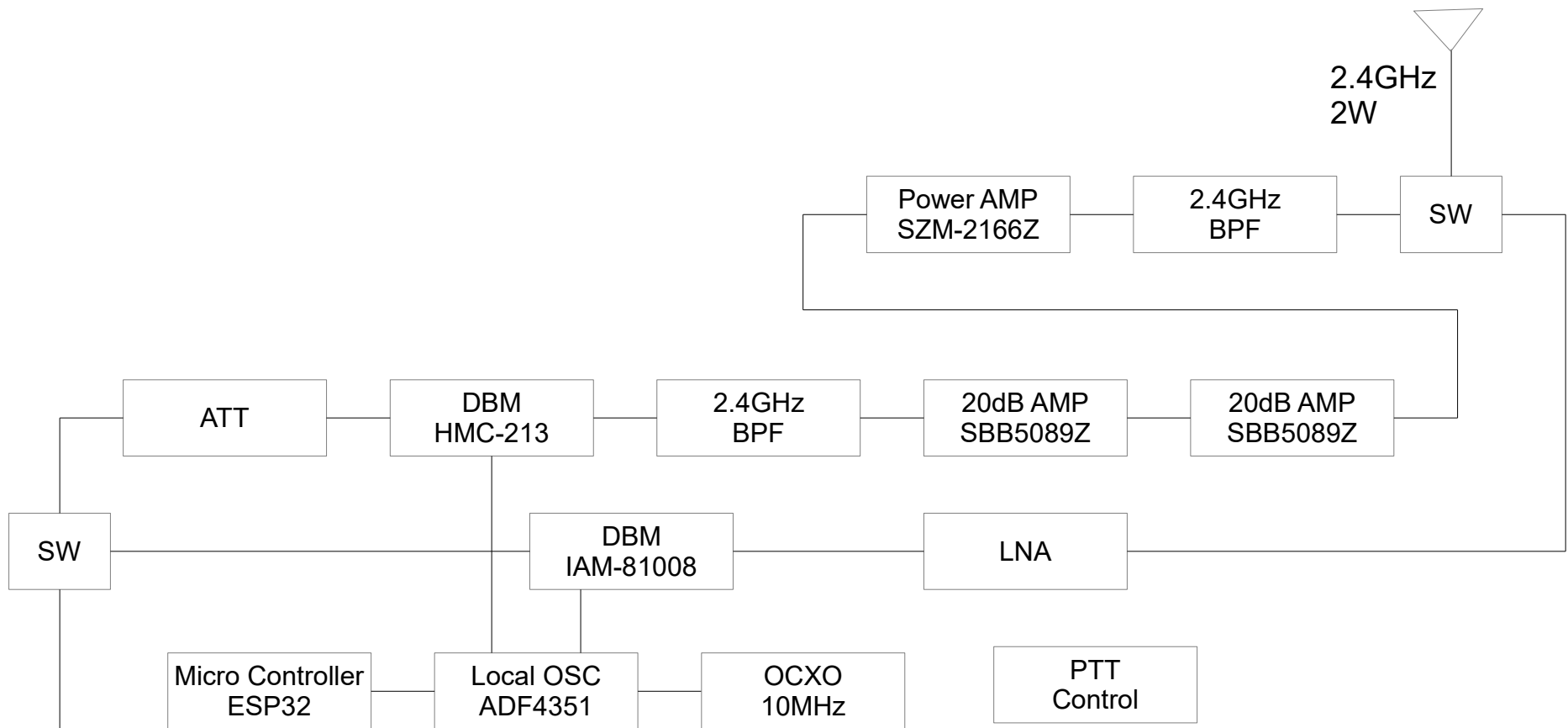
2.4GHzバンドにオンエアしてみよう

従来、マイクロ波用のトランスバータを製作する場合、局発部に多段にわたる通倍回路が必要となります。これが自作を困難にしていると思います。ここでは、ADF4351_PLL基板とマイコン基板を使用して手軽にマイクロ波を発振させることができることを説明します。

1.)2.4GHzトランスバータ ブロックダイアグラム
2.)局発(Local OSC) - 1 PLL基板とマイコン基板について
3.)局発(Local OSC) - 2 ADF4351_PLL基板とESP32結線図
4.)局発(Local OSC) - 3 周波数(レジスタ値)設定ツール
5.)局発(Local OSC) - 4 ソースプログラム 定義部分
6.)局発(Local OSC) - 5 ソースプログラム レジスタ値設定部分
7.)Arduino IDE プログラム統合開発環境
8.)免許申請 二次業務の周波数の使用及び適切な措置について(確認書)

(2024/11/4)担当 JA4CXX / 山本俊通
資料ダウンロード先: www.realvideo.jp

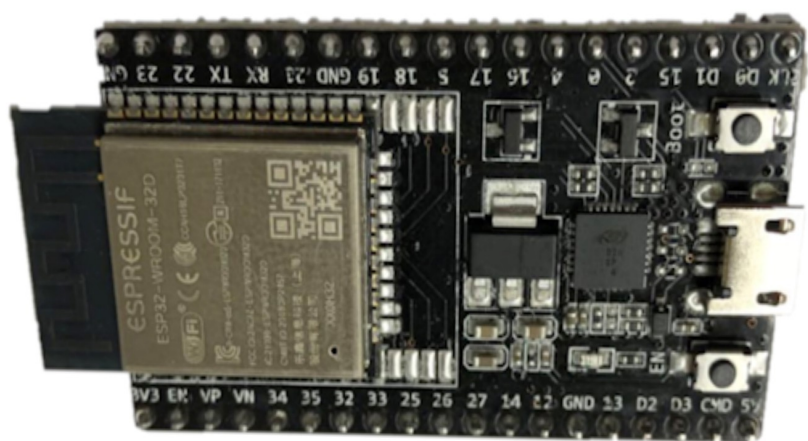
2.4GHz トランスバータ ブロックダイアグラム



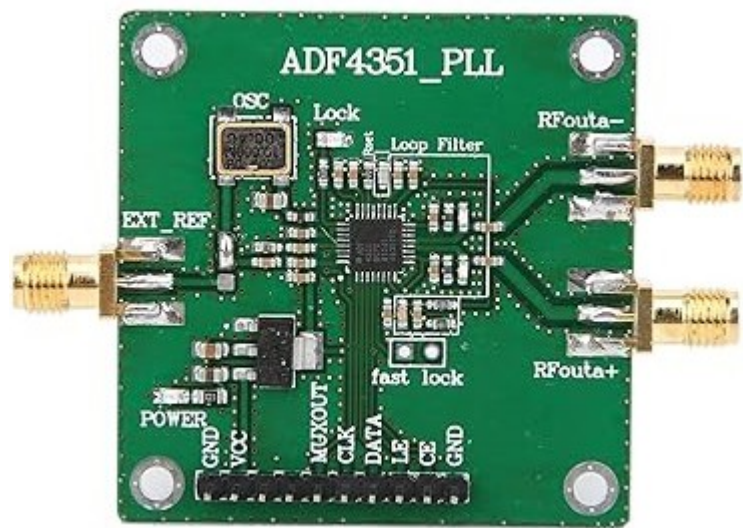
430MHz帯
IC-705



局発(Local OSC) – 1 使用したPLL基板とマイコン基板について

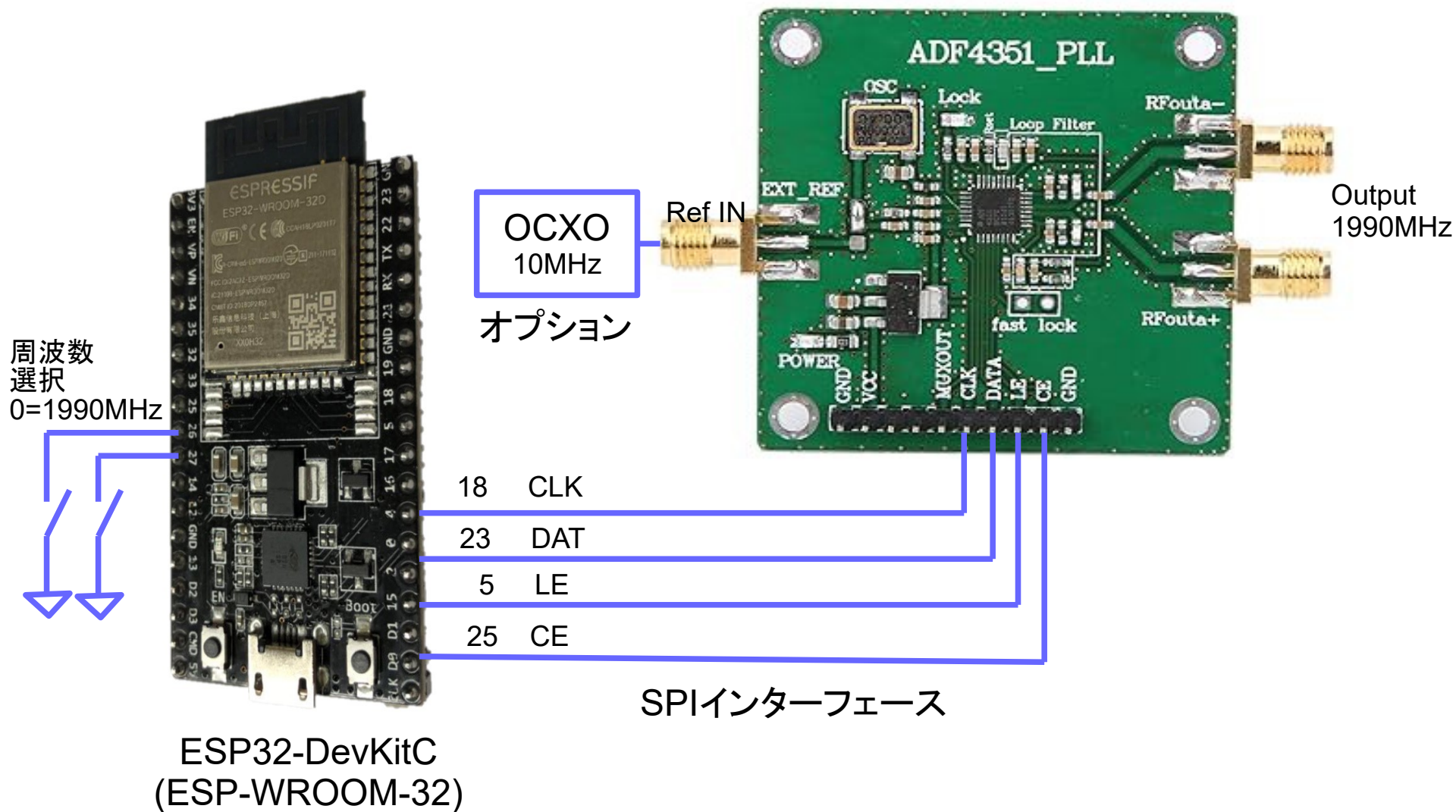


ESP32-DevKitCはESP-WROOM-32と呼ばれるマイコンを搭載したワイヤレスモジュールです。WiFiとBluetoothを搭載したMCUで、動作温度が -40°C から $+125^{\circ}\text{C}$ まで安定した動作が可能な堅牢な設計、超低消費電力設計、さまざまなモジュールを内蔵した高度な統一性を意識した設計などの特徴があります。ESP32マイコンはデュアルコア、動作クロックは240MHz、RAMは520KB、フラッシュメモリは64MBと優れたスペックとなっており、マイコンとしては十分すぎるほどの性能です。



ADF4351は、外部のループ・フィルタと外部からのリファレンス周波数を使うことによって、フラクショナルN(分数分周型)またはインテジャーN(整数分周型)のPLL周波数シンセサイザを実現することができます。ADF4351は、2200MHz~4400MHzの基本出力周波数としての電圧制御発振器(VCO)を内蔵しています。さらに、1/2/4/8/16/32または64の分周回路は、35 MHzまで低いRF出力周波数を発生させることができます。全ての内蔵レジスタの制御は、簡単な3線インターフェースを介して行われます。このデバイスは、3.0V~3.6Vの電源範囲で動作します。

局発(Local OSC) – 2 ADF4351_PLL基板とESP32結線図



局発(Local OSC) – 3 周波数(レジスタ値)設定ツール

ADF4351のレジスタ値決定ツール(ADF435x software)。
レジスタのR5からR0まで順次書き込みます。

The screenshot displays the 'Analog Devices ADF435x Software' application window. The interface is divided into several sections:

- RF Settings:** Includes fields for Output and VCO frequencies (1990 MHz and 3980 MHz), channel spacing (10 kHz), output divider (2), reference frequency (100 MHz), R counter (4), PFD frequency (25 MHz), prescaler (8/9), feedback signal (Fundamenta), and phase adjust (0. Off).
- Register 2:** Configures Low Noise/Spur Mode (Low noise mod), Muxout (3-state output), Double buff (Disabled), Charge pump current (2.50), LDF (FRAC-N), LDP (10 ns), PD Polarity (Positive), Powerdown (Disabled), CP 3-state (Disabled), and Counter reset (Disabled).
- Register 3:** Configures Band Select Clock Mode (Low), Charge Cancellation (Disabled), Clock Divider Value (150), CLK Div Mode (Clock Divider Off), ABP (6 ns (FRAC-N)), and CSR (Disabled).
- Register 4:** Configures VCO Powerdown (Disabled), MTL D (Disabled), Aux Output Select (Divided), Aux Output Enable (0. Disabled), Aux Output Power (-4 dBm), RF Output Enable (1. Enabled), RF Output Power (+5 dBm), and Band Select Clock (Auto set, Divider: 200, Freq (kHz): 125.000).
- Register 5:** Configures LD Pin Mode (Digital Lock Detect).

At the bottom, the 'Registers' section shows a sequence of register addresses and their corresponding values: 0x4F8008, 0x8008029, 0x10E42, 0x4B3, 0x9C803C, and 0x580005. Below these are buttons for 'Write R0' through 'Write R5' and a 'Write All Registers' button.

The status bar at the bottom left indicates 'Application started.' and 'No device connected'. The bottom right corner shows 'Device in use ADF4351 Software version 4.5.0' and the Analog Devices logo.

局発(Local OSC) – 4 ソースプログラム 定義部分

```
////////////////////////////////////
//   ADF4351 RF Generator           /
//   Local-OSC.ino                 /
//   2023-2-16 - 2023-4-4   JA4CXX /
////////////////////////////////////
//   MCU-Pin   ADF4351 Board-Pin
//   18       CLK
//   23       DAT
//   5        LE
//   25       CE
//-----
//   Board Power +5V
////////////////////////////////////
// IO26 = Bit0 pinMode(26, INPUT_PULLUP);
// IO27 = Bit1 pinMode(27, INPUT_PULLUP);
// Case 0 = 1990MHz
// Case 1 = 1980MHz
// Case 2 = 1970MHz
// Case 3 = 1960MHz
////////////////////////////////////

#include "ADF4351.h"
#include <SPI.h>

#define clock 18
#define data 23
#define LE 5
#define CE 25

ADF4351 adf4351(clock,data,LE,CE); // declares object PLL of type ADF4351

void setup() {

    adf4351.begin();

    pinMode(26, INPUT_PULLUP); // 入力ピン(プルアップ抵抗付)
    pinMode(27, INPUT_PULLUP); // 入力ピン(プルアップ抵抗付)
```

局発(Local OSC) – 5 ソースプログラム レジスタ値設定部分

```
int CHN = 0;
int SW0 = digitalRead(26);
int SW1 = digitalRead(27);

CHN = SW0 + SW1 * 2;

switch (CHN) {
  case 0: //1990MHz
    adf4351.WriteRegister(0x00580005); // -R5- default value LD working mode
    adf4351.WriteRegister(0x9C803C); // -R4- output divider = 16 // band select clock divider
    adf4351.WriteRegister(0x000004B3); // -R3- default Antibacklash pulse width
    adf4351.WriteRegister(0x00010E42); // -R2- Low noise mode, R Counter=4
    adf4351.WriteRegister(0x8008029); // -R1- prescaler=4/5、MOD=25
    adf4351.WriteRegister(0x4F8008); // -R0- int = 128 frac = 0

    break;

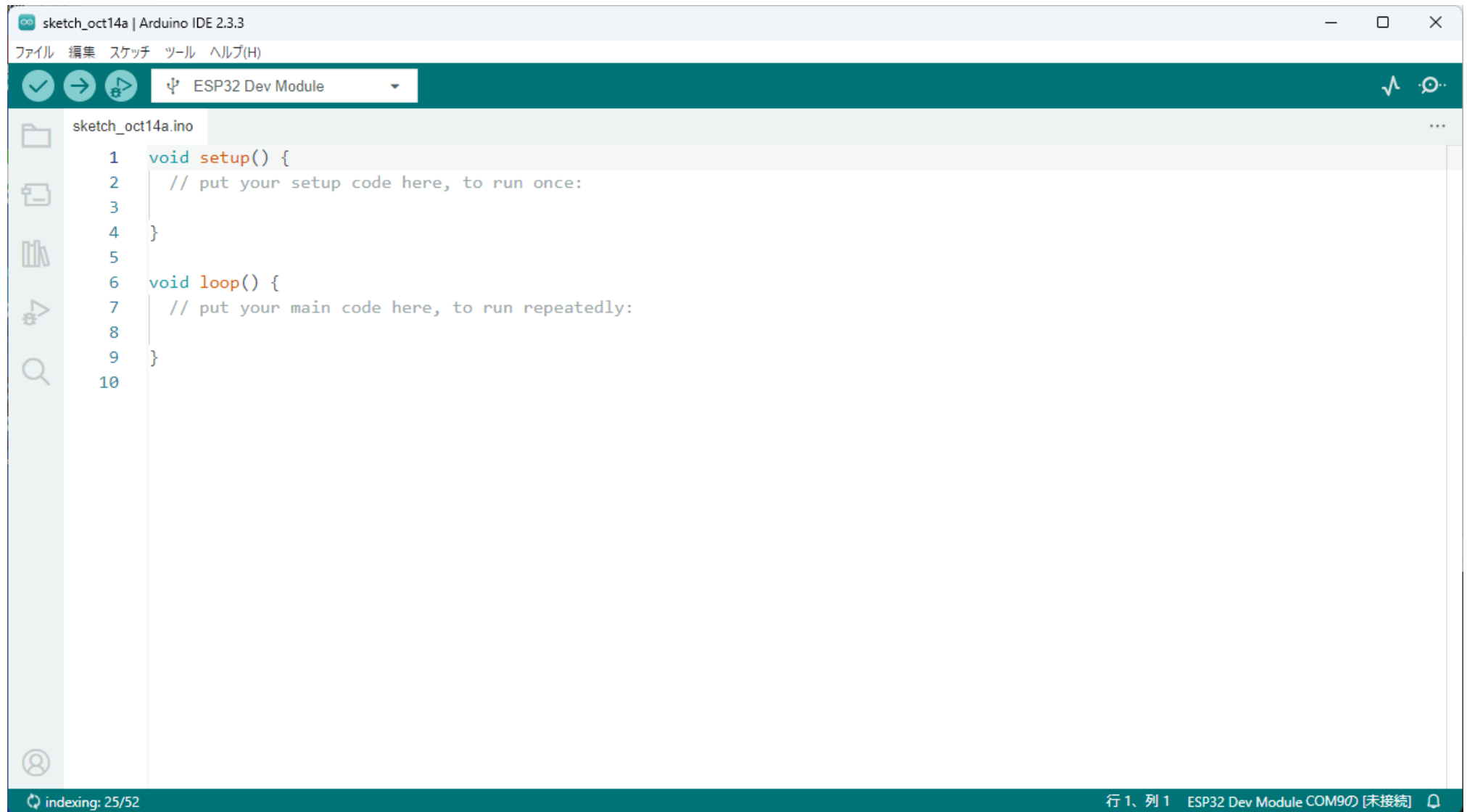
  ~~~~~

  ~~~~~

  default:
    // どのcaseにも一致しなかったとき実行される
    // (defaultは省略可能)
    break;
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

Arduino IDE プログラム統合開発環境



起動時

Arduino IDE プログラム統合開発環境 — ボードとポートを選択

The screenshot shows the Arduino IDE 2.3.3 interface. The top menu bar includes 'ファイル', '編集', 'スケッチ', 'ツール', and 'ヘルプ(H)'. The 'Tools' menu is open, showing 'Arduino Nano' selected. Below it, 'Arduino Nano COM9' and '不明 COM4' are listed. A red box highlights the option '他のボードとポートを選択...'. A dialog box titled '他のボードとポートを選択' is open, displaying a list of boards and ports. The board 'ESP32 Dev Module' is selected, and the port 'COM9 Serial Port (USB)' is selected. The dialog box includes instructions in Japanese and buttons for 'キャンセル' and 'OK(O)'. The background shows a sketch with code for an ESP32 module, including pin definitions and SPI library includes. The output window at the bottom shows the upload progress and completion message.

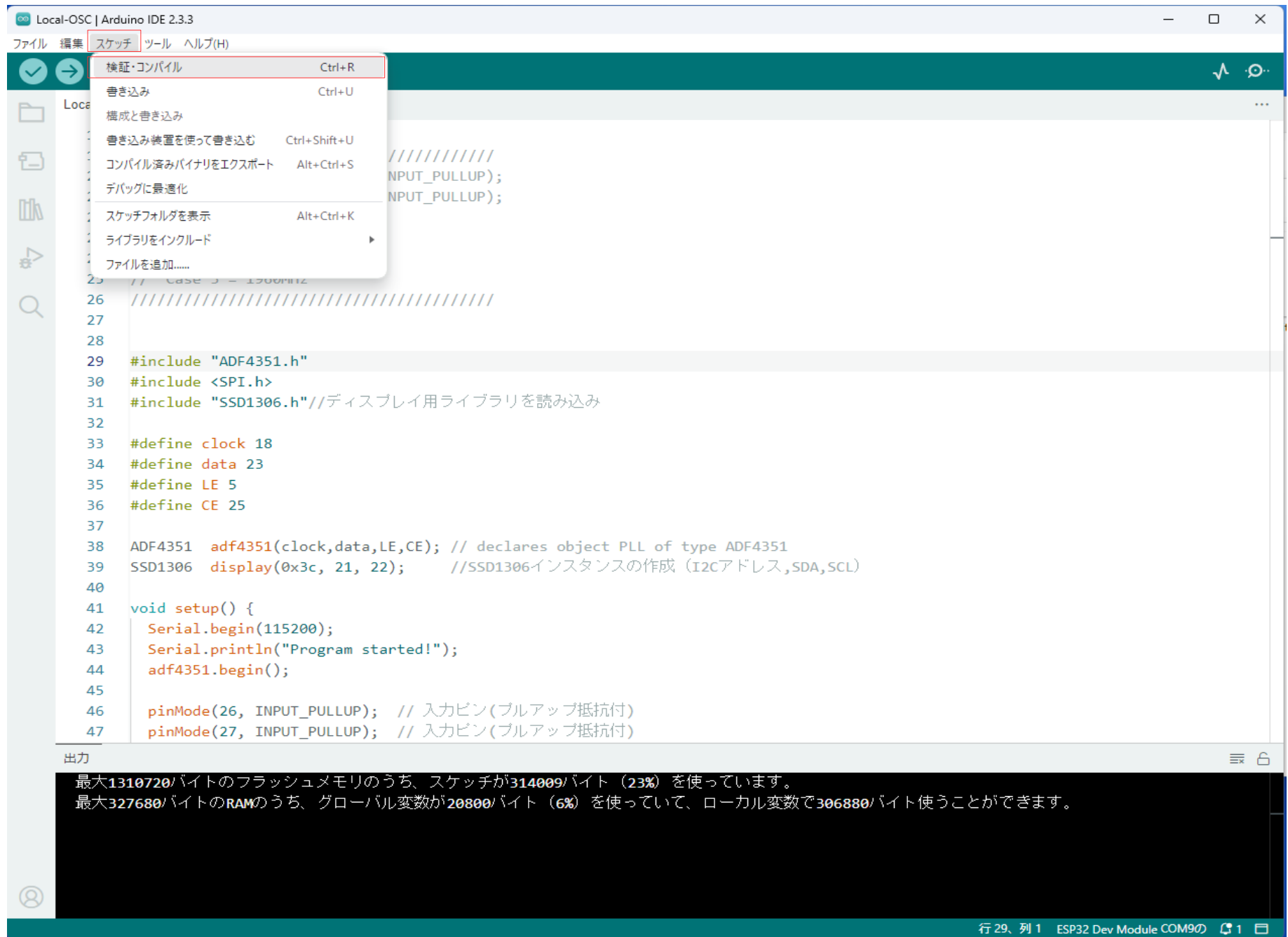
```
1 // 23 DAT
2 // 5 LE
3 // 25 CE
4 // -----
5 // MCU-Pin SSD1306
6 // 21 SDA
7 // 22 SCL
8 // -----
9 // Board Power +5V
10 // -----
11 // IO26 = Bit0 pin
12 // IO27 = Bit1 pin
13 // Case 0 = 1990MHz
14 // Case 1 = 1980MHz
15 // Case 2 = 1970MHz
16 // Case 3 = 1960MHz
17 // -----
18 #include "ADF4351.h"
19 #include <SPI.h>
```

出力

```
Writing at 0x000585ab... (100 %)
Wrote 314368 bytes (176912 compressed) at 0x00010000 in 3.1 seconds (effective 815.2 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Arduino IDE — プログラムをコンパイルしてバグ潰し



Local-OSC | Arduino IDE 2.3.3

ファイル 編集 **スケッチ** ツール ヘルプ(H)

- 検証・コンパイル Ctrl+R
- 書き込み Ctrl+U
- 構成と書き込み
- 書き込み装置を使って書き込む Ctrl+Shift+U
- コンパイル済みバイナリをエクスポート Alt+Ctrl+S
- デバッグに最適化
- スケッチフォルダを表示 Alt+Ctrl+K
- ライブラリをインクルード
- ファイルを追加.....

```
25 // case 3 - 1500MHz
26 ///////////////////////////////////////////////////
27
28
29 #include "ADF4351.h"
30 #include <SPI.h>
31 #include "SSD1306.h"//ディスプレイ用ライブラリを読み込み
32
33 #define clock 18
34 #define data 23
35 #define LE 5
36 #define CE 25
37
38 ADF4351 adf4351(clock,data,LE,CE); // declares object PLL of type ADF4351
39 SSD1306 display(0x3c, 21, 22); //SSD1306インスタンスの作成 (I2Cアドレス,SDA,SCL)
40
41 void setup() {
42   Serial.begin(115200);
43   Serial.println("Program started!");
44   adf4351.begin();
45
46   pinMode(26, INPUT_PULLUP); // 入力ピン(プルアップ抵抗付)
47   pinMode(27, INPUT_PULLUP); // 入力ピン(プルアップ抵抗付)
```

出力

最大1310720バイトのフラッシュメモリのうち、スケッチが314009バイト (23%) を使っています。
最大327680バイトのRAMのうち、グローバル変数が20800バイト (6%) を使っていて、ローカル変数で306880バイト使うことができます。

行 29、列 1 ESP32 Dev Module COM9の 1

Arduino IDE — プログラムを基板に書き込みます

The screenshot displays the Arduino IDE interface. The top menu bar includes 'ファイル', '編集', 'スケッチ', 'ツール', and 'ヘルプ(H)'. The toolbar shows icons for '検証・コンパイル' (Verify/Compile) and '書き込み' (Upload). The 'ESP32 Dev Module' is selected in the board manager dropdown. The main editor shows a sketch named 'Local-OSC.ino' with the following code:

```
1 ////////////////////////////////////////////////////
2 // ADF4351 RF Generator /
3 // Local-OSC.ino /
4 // 2023-2-16 - 2023-4-4 JA4CXX /
5 ////////////////////////////////////////////////////
6 // MCU-Pin ADF4351 Board-Pin
7 // 18 CLK
8 // 23 DAT
9 // 5 LE
10 // 25 CE
11 //-----
12 // Board Power +5V
13 //-----
14 // MCU-Pin SSD1306 Board-Pin
15 // 21 SDA
16 // 22 SCL
17 //-----
18 // Board Power +3.3V
19 //-----
20 // IO26 = Bit0 pinMode(26, INPUT_PULLUP);
21 // IO27 = Bit1 pinMode(27, INPUT_PULLUP);
22 // Case 0 = 1990MHz
23 // Case 1 = 1980MHz
24 // Case 2 = 1970MHz
25 // Case 3 = 1960MHz
26 //-----
27
28
29 #include "ADF4351.h"
30 #include <SPI.h>
```

A context menu is open over the sketch, listing the following options:

- 検証・コンパイル (Ctrl+R)
- 書き込み (Ctrl+U)
- 構成と書き込み
- 書き込み装置を使って書き込む (Ctrl+Shift+U)
- コンパイル済みバイナリをエクスポート (Alt+Ctrl+S)
- デバッグに最適化
- スケッチフォルダを表示 (Alt+Ctrl+K)
- ライブラリをインクルード
- ファイルを追加.....

The output window at the bottom shows the following text:

```
出力
Writing at 0x000585ab... (100 %)
Wrote 314368 bytes (176912 compressed) at 0x00010000 in 3.1 seconds (effective 815.2 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

A notification box in the bottom right corner indicates '書き込み完了' (Upload completed).

二次業務の周波数の使用及び適切な措置について（確認書）

2024年 1月 12日

中国総合通信局長（注1）

申請（届出）に係るアマチュア局の二次業務の周波数（2,425MHz帯及び5,750MHz帯）の使用に当たっては、一次業務の無線局に有害な混信を生じさせることがないように、下記のとおり適切な措置を執るものとします。なお、一次業務の無線局からの有害な混信に対して保護は要求しません。

1. 申請者（届出者）及び申請（届出）に係るアマチュア局に関する事項（注2）

① 氏名又は名称及び代表者氏名	ヤマモトシミヂ 山本俊通
② 呼出符号	JA4CXX
③ 免許の番号	中A第XXXXX号

2. 二次業務の周波数の確認及び適切な措置の方法

①二次業務の周波数の確認（注3）

2,425MHz帯（2,400MHzから2,450MHzまで）及び5,750MHz帯（5,650MHzから5,850MHzまで）の周波数は、二次業務の周波数であることを確認しました。

②適切な措置の方法（注4）

申請（届出）に係るアマチュア局が発射する電波が一次業務の無線局に有害な混信を与えることがないように、次の事項について確約します。

電波を発射しようとする前に、当該周波数及びその近接する周波数が使用されていないことを確認します。

電波を発射しようとする前に、電波の送信の地点の周囲の状況等を確認します。

（例）・免許状記載事項（附款を含む。）の条件に適合する使用に限る。

・免許人が所有又は管理する一の構内や屋内での使用に限る。

・住宅地外での使用に限る。

発射する電波の空中線電力は、通信を行うため必要最小のものとしします。

一次業務の無線局に有害な混信を与える恐れがあるときは、直ちに電波の発射を停止します。

一次業務の無線局又は総合通信局長（沖縄総合通信事務所長を含む。以下同じ。）から電波の発射の停止を求められた場合は、直ちに電波の発射を停止します。

総合通信局長から、適切な措置について詳細な資料の提出を求められた場合は、速やかにその資料を提出します。

（注）一次業務の無線局に有害な混信を生じさせた場合、処罰の対象となります。適切な措置の方法を執った上で、十分に注意してアマチュア局を運用してください。

3. その他（注5）

注1 所轄総合通信局長を記載すること。なお、沖縄県の区域においては、沖縄総合通信事務所長とする。

注2 1の欄は、次によること。

(1) ②の欄は、申請（届出）に係るアマチュア局に指定されている呼出符号を記載すること。

(2) ③の欄は、申請（届出）に係るアマチュア局の免許の番号を記載すること。

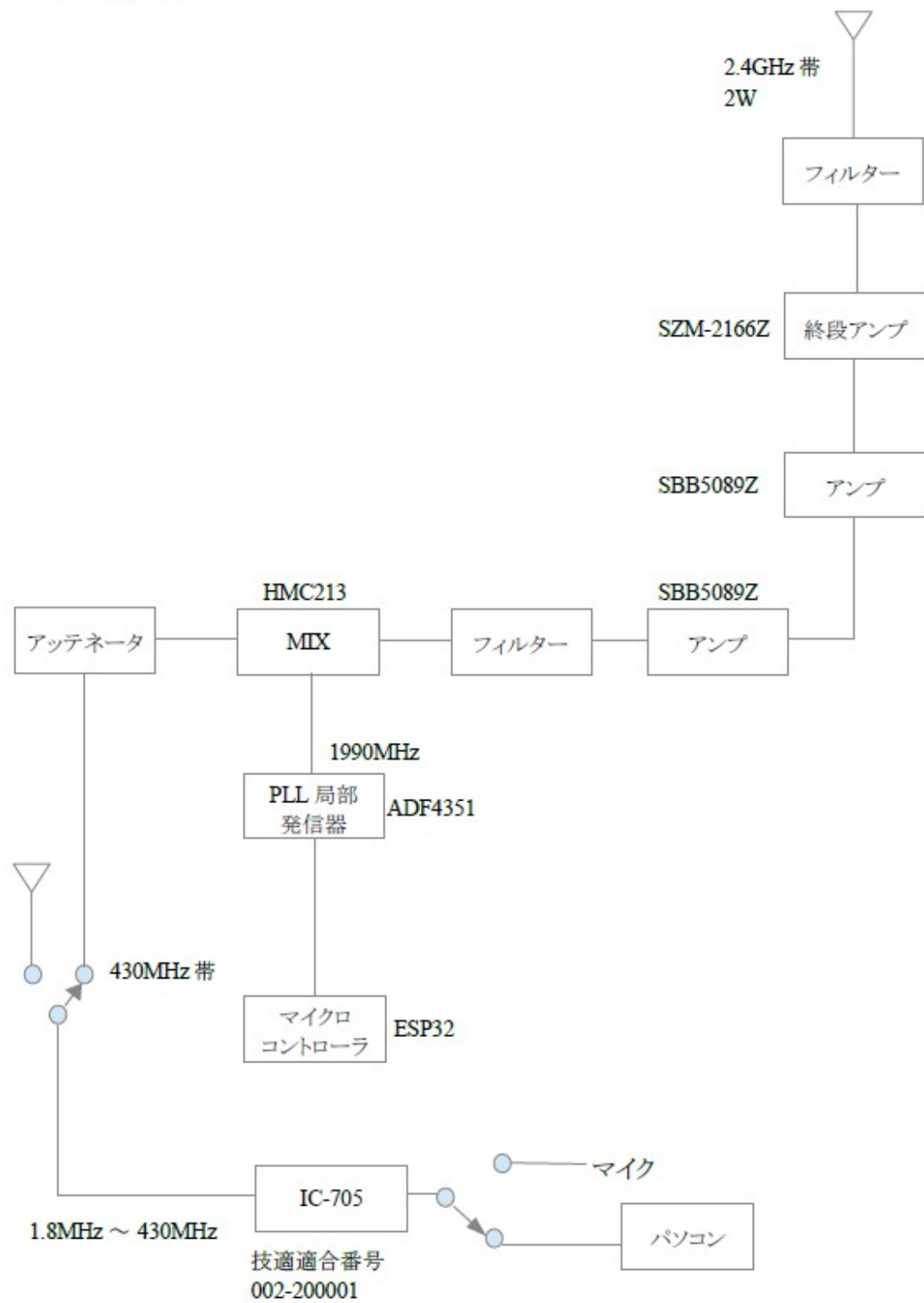
注3 2,425MHz帯及び5,750MHz帯の周波数が二次業務の周波数であることを確認したときは、該当する口にレ印を付けること。

注4 適切な措置の方法として確約する事項について、該当する口にレ印を付けること。

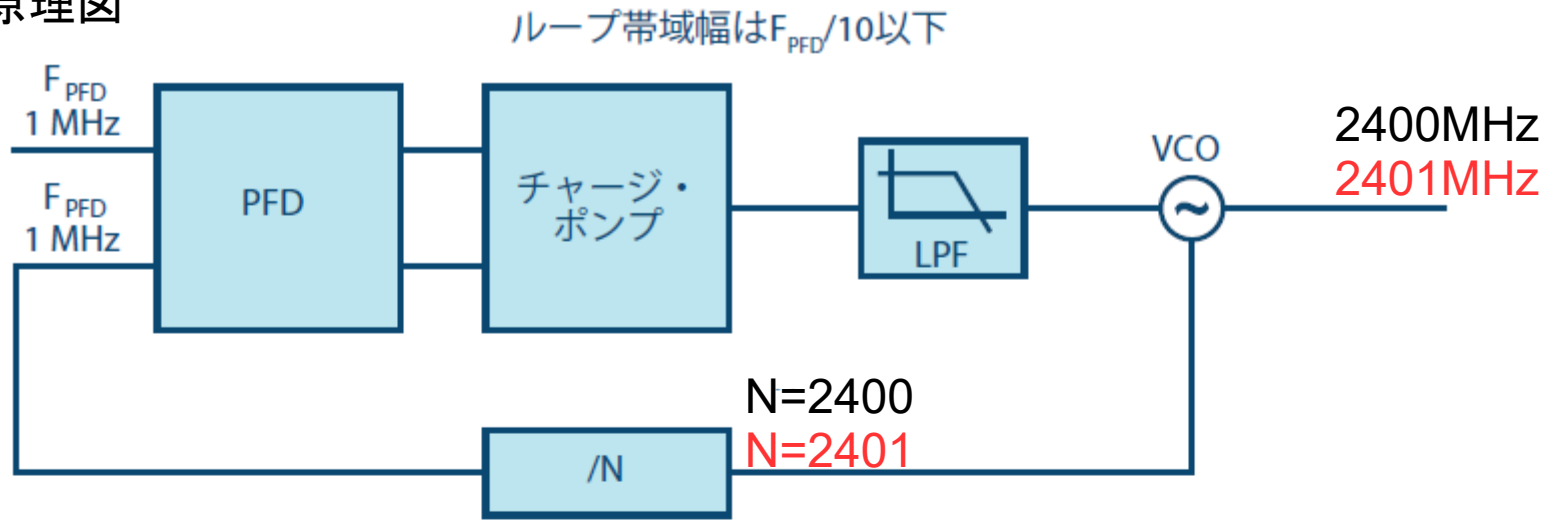
注5 その他必要な事項がある場合は、その内容について記載すること。

8.) 免許申請
確認書の各確認事項を
確認してチェック欄にレ点を
入れて提出します。

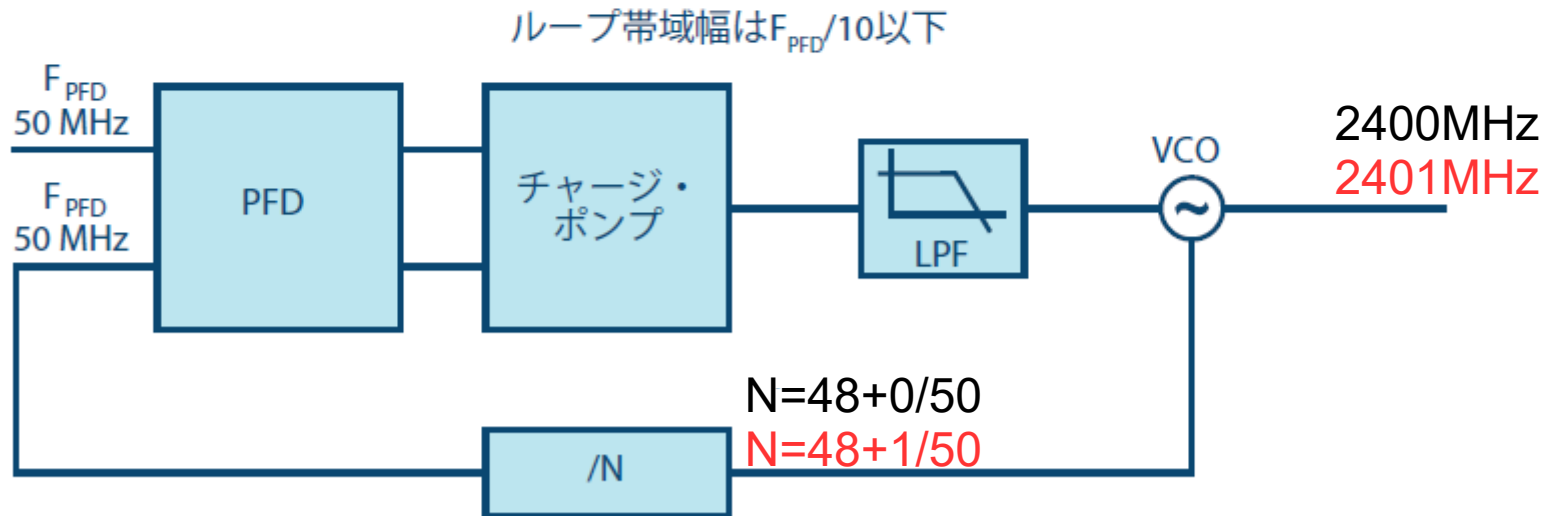
免許申請に使用した 送信機系統図



参考 PLL原理図



インテジャーN型のPLL



フラクショナルN型のPLL