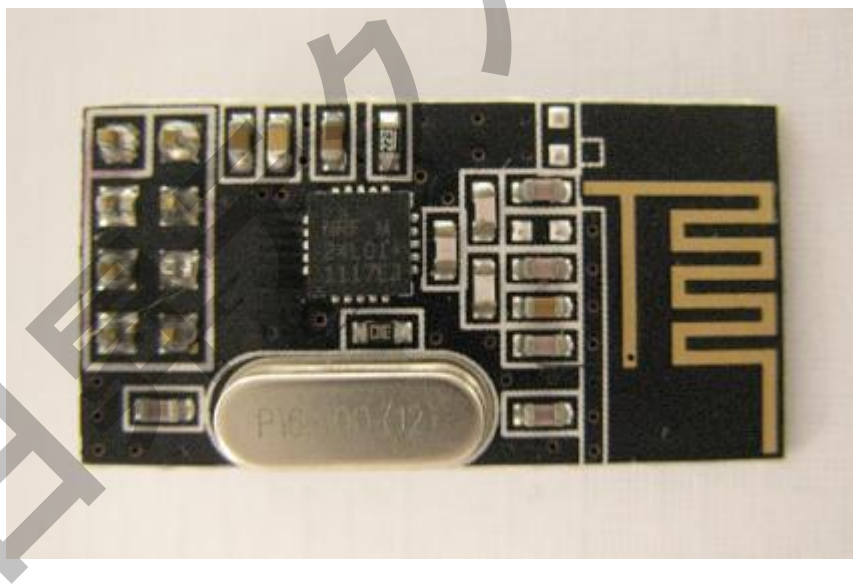
無線モジュール NRF24L01 の MCU 8051 向けユーザーマニュアル

株式会社日昇テクノロジー

http://www.csun.co.jp

info@csun.co.jp

更新日 2013/09/10



修正履歴

NO	バージョン	修正内容	修正日
1	Ver1.0	新規作成	2013/09/10

※ この文書の情報は、文書を改善するため、事前の通知なく変更されることがあります。最新版は弊社ホームページからご参照ください。http://www.csun.co.jp

目次

チップ概要

NRF24L01 は NORDIC 会社生産した無線通信用のチップで、FSK 変調を使用し、内部に NORDIC 社の Enhanced Short Burst プロトコルがパッケージされる。ポイント･ツー･ポイント又は１ VS 6 の無線通信を実現できる。

無線通信速度は最大 2M（BPS）まで、NORDIC 社は通信モジュールの GERBER ファイルを提供し、 直接処理・生産可能。組み込みエンジニアや SCM 愛好家は SCM システムの取り置きの５つの GPIO と１つの割り込み入力ピンを使用し、無線通信機能を実現でき、 MCU システムの無線機能構築には非常に便利となる。

1、 NRF24L01 機能ブロック図

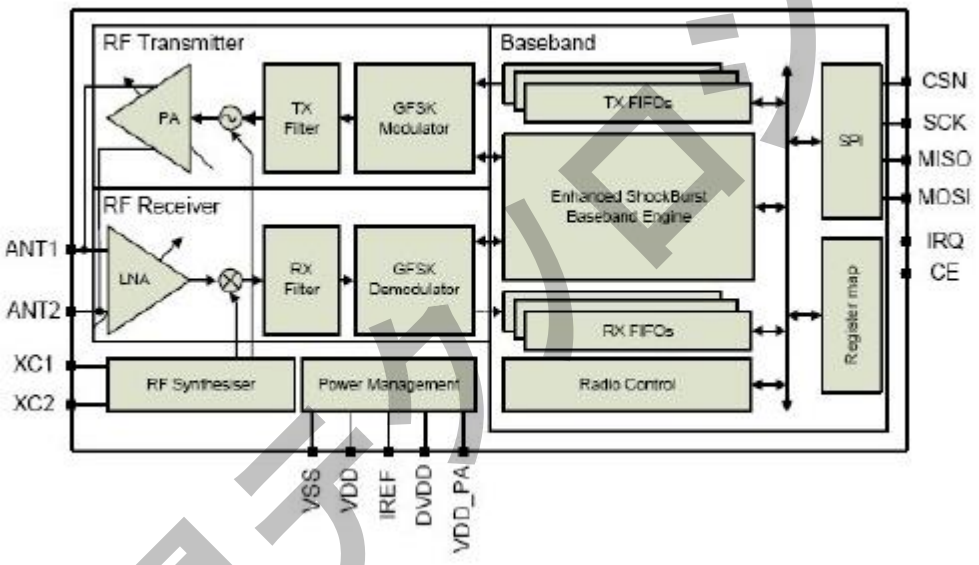


Fig.1 NRF24L01 BLOCK DIAGRAM

NRF24L01 機能ブロック図はFig.1の示す通り、Fig.1右側の６つの制御とデータ信号は順次にCSN､SCK､MISO､MOSI､IRQ､CE である。

- CSN：チップセレクトライン、ローレベルチップ動作機能。
- SCK：チップ制御のクロックライン（SPI クロック）
- MISO：チップ制御のデータライン（Master input slave output）
- MOSI：チップ制御のデータライン（Master output slave input）
- IRQ：割り込み信号。無線通信中 MCU は IRQ を介し NRF24L01 と通信する。
- CE： チップのモード制御ライン。 CSN はローレベルの場合、CE と NRF24L01 の CONFIG レジスタは NRF24L01 のステータスを合わせて決定する（ NRF24L01 のステートマシンを参照)。

2、 NRF24L01 ステートマシン

NRF24L01 の ステートマシンは Fig.2 を参照。NRF24L01 ファームウェアのプログラミングはステートマシンに基づき動作する。ステータスは下記の通り：

- Power Down Mode ：パワーダウンモード
- Tx Mode ：送信モード
- Rx Mode ：受信モード
- Standby-1Mode ：スタンバイ 1 モード
- Standby-2Mode ：スタンバイ 2 モード

上記の 5 つのモードの切り替え方法と必要時間は Fig.2 を参照。

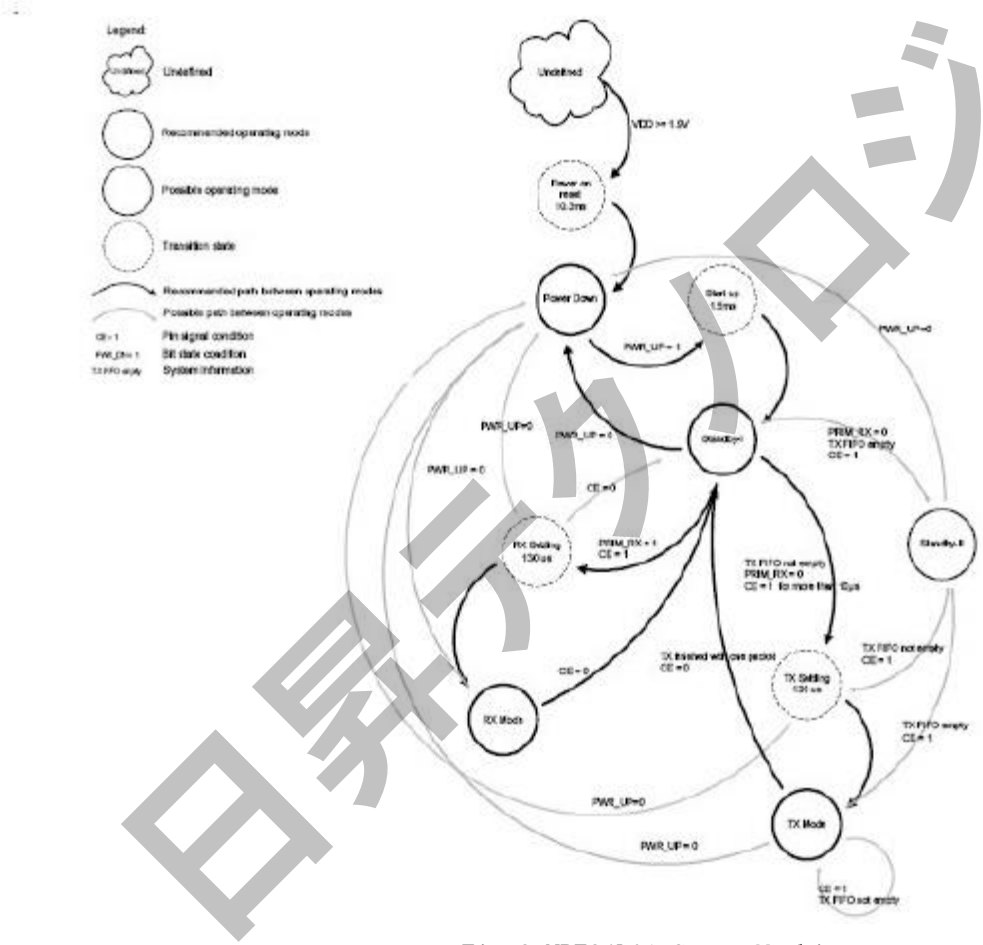


Fig.2 NRF24L01 State Machine

24L01 ファームウェアプログラミングの基本プロセスは下記の通り：

1) CSN を低レベルに設定、チップ有効にする、チップの各パラメータの設定。(3.Tx と Rx 設定プロセスを参照)。設定は Power Down 状態で行う。

2) Tx モードの場合、 Tx FIFO にデータを書込む。

3) 設定完了後、 CE と CONFIG 中の PWR_UP/ PRIM_RX パラメータに基づき 24L01 の切り替えステータスを決定する。

Tx Mode：PWR_UP=1； PRIM_RX=0； CE=1 (10US 以上を確保)；

Rx Mode:　PWR_UP=1;　PRIM_RX=1;　CE=1;

4) IRQ ピンは、次の 3 つの場合で低レベルになる：
Tx FIFO　送信完了、ACK を受信（ ACK 有効の場合）
Rx FIFO　データ受信
最大再送回数に達した
IRQ を外部割り込み入力ピンに接続、割り込みプログラムで処理する。

3、 Tx と Rx の設定プロセス

本節は ENHANCED SHORT BURST 通信方式を使用する Tx と Rx の設定と通信プロセスを説明する。

3.1 Tx　モード初期化プロセス

初期化ステップ	24L01 関連レジスタ
1) Tx ノードのアドレス書き込み	TX_ADDR
2) Rx ノードのアドレス書き込み（ Auto Ack 有効するため）	RX_ADDR_P0
3) AUTO ACK 有効する	EN_AA
4) PIPE 0 有効する	EN_RXADDR
5) 自動再送回数設定	SETUP_RETR
6) 通信周波数選択	RF_CH
7) 送信パラメータ設定（低雑音アンプゲイン、送信パワー、ワイヤレススピード）	RF_SETUP
8) チャネル 0 有効データ幅選択	Rx_Pw_P0
9) 24L01 基本パラメータとモード切り替え設定	CONFIG

3.2 Rx モード初期化プロセス

初期化ステップ	24L01 関連レジスタ
1) Rx ノードのアドレス書き込み	RX_ADDR_P0
2) AUTO ACK 有効する	EN_AA
3) PIPE 0 有効する	EN_RXADDR
4) 通信周波数選択	RF_CH
5) チャネル 0 有効データ幅選択	Rx_Pw_P0
6) 送信パラメータ設定（低ノイズアンプゲイン、送信パワー、ワイヤレススピード）	RF_SETUP
7) 24L01 基本パラメータとモード切り替え設定	CONFIG

4、 制御プログラム説明

4.1 関数説明

NRF24L01 の制御プログラムは下記の関数が含まれる：

```
uchar SPI_RW(uchar byte);
uchar SPI_RW_Reg(uchar reg、 uchar value);
uchar SPI_Read(uchar reg);
uchar SPI_Read_Buf(uchar reg、 uchar *pBuf、 uchar bytes);
uchar SPI_Write_Buf(uchar reg、 uchar *pBuf、 uchar bytes);
void RX_Mode(void);
void TX_Mode(void);
```

4.1.1 uchar SPI_RW(uchar byte)

```
uchar SPI_RW(uchar byte)
{
    uchar bit_ctr;
    for(bit_ctr=0;bit_ctr<8;bit_ctr++)  // output 8-bit
    {
        MOSI = (byte & 0x80);  // output 'byte'、 MSB to MOSI
        byte = (byte << 1); // shift next bit into MSB..
        SCK = 1;    // Set SCK high..
        byte |= MISO;   // capture current MISO bit
        SCK = 0;    // ..then set SCK low again
    }
    return(byte);   // return read byte
}
```

基本関数では GPIO から SPI 機能を模擬し、出力バイト（MOSI）を MSB から繰り返し出力し、入力バイト（MISO）を LSB から循環にシフトする。立ち上がりエッジで読み込み、立ち下がりエッジで出力する（SCK の初期化はローレベル)。

4.1.2 uchar SPI_RW_Reg (uchar reg、 uchar value)

```
uchar SPI_RW_Reg(uchar reg、 uchar value)
{
      uchar status;
      CSN = 0;    // CSN low、 init SPI transaction
      status = SPI_RW(reg);   // select register
      SPI_RW(value);  // ..and write value to it..
      CSN = 1;    // CSN high again
      return(status); // return nRF24L01 status byte
}
```

レジスタアクセス関数：24L01 レジスタの値を設定する機能。WRITE_REG コマンド（0x20+レジスタアドレス）を使用し、設定値を対応レジスタに書き込む、フィードバック値を読み取る。関数機能から見て、value の値を reg レジスタに書き込むことである。

注意するのは：NRF24L01 にアクセスする前に、チップ有効を（CSN=0；）し、完了後チップ無効（CSN=1；）動作は必要とする。

4.1.3 uchar SPI_Read (uchar reg);

```
uchar SPI_Read(uchar reg)
{
      uchar reg_val;
      CSN = 0;    // CSN low、 initialize SPI communication...
      SPI_RW(reg);    // Select register to read from..
```

```
    reg_val = SPI_RW(0);    // ..then read registervalue

    CSN = 1;    // CSN high、 terminate SPI communication

    return(reg_val);    // return register value

}
```

レジスタ読み取り関数： READ_REG コマンド（0x00+レジスタアドレス）を使用し、レジスタの値を読み出す。関数機能から見て、reg レジスタの値を reg_val に読み取ることである。

4.1.4 uchar SPI_Read_Buf (uchar reg、 uchar *pBuf、 uchar bytes);

```
uchar SPI_Read_Buf(uchar reg、 uchar *pBuf、 uchar bytes)

{

    uchar status、byte_ctr;

    CSN = 0;    // Set CSN low、 init SPI tranaction

    status = SPI_RW(reg);   // Select register to write to and read status byte

    for(byte_ctr=0;byte_ctr<bytes;byte_ctr++)

        pBuf[byte_ctr] = SPI_RW(0); // Perform SPI_RW to read byte from nRF24L01

    CSN = 1;    // Set CSN high again

    return(status); // return nRF24L01 status byte

}
```

バッファ受信アクセス関数：受信時に FIFO バッファの値を読み取る機能。READ_REG コマンドで、FIFO（RD_RX_PLOAD）からデータを読み取り、配列に保存する。

4.1.5 uchar SPI_Write_Buf (uchar reg、 uchar *pBuf、 uchar bytes);

```
uchar  SPI_Write_Buf(uchar  reg、  uchar  *pBuf、  uchar  bytes)

{
```

```
uchar  status、byte_ctr;

      CSN  =  0;  //  Set  CSN  low、   init  SPI  tranaction
      status  =  SPI_RW(reg); //  Select  register  to  write  to  and  read  status  byte
      Uart_Delay(10);
      for(byte_ctr=0; byte_ctr<bytes; byte_ctr++) // then write all byte in buffer(*pBuf)
      SPI_RW(*pBuf++);
      CSN  =  1;  //  Set  CSN  high  again
      return(status); //  return  nRF24L01  status  byte
}
```

バッファ送信アクセス関数：配列内のデータを FIFO バッファに送信する機能。WRITE_REG コマンドで、データを FIFO（WR_TX_PLOAD）に書き込むこと。

4.1.6 void RX_Mode(void)

24L01 を受信モードに設定、プロセスは 3.2 Rx 初期化を参照。

```
void  RX_Mode(void)
{
CE=0;
SPI_Write_Buf(WRITE_REG  +  RX_ADDR_P0、  TX_ADDRESS、  TX_ADR_WIDTH);
SPI_RW_Reg(WRITE_REG  +  EN_AA、  0x01);  //  Enable  Auto.Ack:Pipe0
SPI_RW_Reg(WRITE_REG  +  EN_RXADDR、  0x01);  //  Enable  Pipe0
SPI_RW_Reg(WRITE_REG  +  RF_CH、  40);  //  Select  RF  channel  40
SPI_RW_Reg(WRITE_REG  +  RX_PW_P0、  TX_PLOAD_WIDTH);
SPI_RW_Reg(WRITE_REG  +  RF_SETUP、  0x07);
SPI_RW_Reg(WRITE_REG  +  CONFIG、  0x0f); //  Set  PWR_UP  bit、  enable  CRC(2  bytes)
&  Prim:RX.  RX_DR  enabled..

CE  =  1;  //  Set  CE  pin  high  to  enable  RX  device

//    This device is now ready to receive one packet of 16 bytes payload from a TX device
sending  to  address
//    '3443101001'、  with auto acknowledgment、  retransmit count of 10、  RF channel 40 and
datarate  =  2Mbps.

}
```

4.1.7 void TX_Mode(void)

24L01 を送信モードに設定、プロセスは 3.1 Tx 初期化を参照。

```
void  TX_Mode(void)
```

```
{
    CE=0;

    SPI_Write_Buf(WRITE_REG + TX_ADDR、 TX_ADDRESS、 TX_ADR_WIDTH);
    SPI_Write_Buf(WRITE_REG + RX_ADDR_P0、 TX_ADDRESS、 TX_ADR_WIDTH);
    SPI_Write_Buf(WR_TX_PLOAD、 tx_buf、 TX_PLOAD_WIDTH); // Writes data to TX payl
oad

    SPI_RW_Reg(WRITE_REG + EN_AA、 0x01); // Enable Auto.Ack:Pipe0
    SPI_RW_Reg(WRITE_REG + EN_RXADDR、 0x01); // Enable Pipe0
    SPI_RW_Reg(WRITE_REG + SETUP_RETR、 0x1a); // 500us + 86us、 10 retrans...
    SPI_RW_Reg(WRITE_REG + RF_CH、 40); // Select RF channel 40
    SPI_RW_Reg(WRITE_REG + RF_SETUP、 0x07); // TX_PWR:0dBm、 Datarate:2Mbps、
    LNA:HCURR
    SPI_RW_Reg(WRITE_REG + CONFIG、 0x0e);
  // Set PWR_UP bit、 enable CRC(2 bytes)
    & Prim:TX. MAX_RT & TX_DS enabled..
    CE=1;

}
```

4.2 NRF24L01 関連コマンドのマクロ定義

nRF24L01 は固定のタイミングシーケンスとコマンドによって、チップの受送信を制御する。制御コマンドは FIG..4.2.1 の示す通り。

SPI インタフェースコマンド		
コマンド	**フォーマット**	**説明**
R_REGISTER	000AAAAA	設定レジスタ読み取り。AAAAA は読み取りレジスタアドレス。
W_ REGISTER	001AAAAA	設定レジスタ書き込み。AAAAA は書き込みレジスタアドレス。(電源オフ/待機モード)
R_RX_PAYLOAD	01100001	RX 有効データ読み取り：1-32 バイト。読み取り動作はバイト 0 から。 RX 有効データ読み取り完了後、FIFO レジスタクリア。(受信モード)
W_ RX_PAYLOAD	10100000	RX 有効データ書き込み：1-32 バイト。書き込み動作はバイト 0 から。(送信モード)
FLUSH_TX	11100001	TX_FIFO レジスタクリア。(送信モード)
FLUSH_RX	11100010	RX_FIFO レジスタクリア。(受信モード) 応答信号を送信する処理でこのコマンドを実行しない（実行すると、応答信号は完全送信できない)
REUSE_TX_PL	11100011	(送信ノード) 前の送信パケットの有効データを再利用。CE＝１の場合、データは常に再送信される。データパケットを送信する時、パケット再利用機能を禁止する必要がある。
NOP	11111111	動作なし。ステータスレジスタを読み取るために使用する。

FIG.4.2.1

この前使用される関数も下記のコマンドが必要とする：

```
SPI_RW_Reg(WRITE_REG + EN_RXADDR、 0x01);
SPI_Write_Buf(WRITE_REG + TX_ADDR、 TX_ADDRESS、 TX_ADR_WIDTH);
```

関連コマンドのマクロ定義は下記の通り：

```
#define  READ_REG     0x00    //  Define  read  command  to  register
#define  WRITE_REG    0x20    //  Define  write  command  to  register
#define  RD_RX_PLOAD  0x61    //  Define  RX  payload  register  address
#define  WR_TX_PLOAD  0xA0    //  Define  TX  payload  register  address
#define  FLUSH_TX     0xE1    //  Define  flush  TX  register  command
#define  FLUSH_RX     0xE2    //  Define  flush  RX  register  command
#define  REUSE_TX_PL  0xE3    //  Define  reuse  TX  payload  register  command
#define  NOP          0xFF   //  Define  No  Operation  、    might  be  used  to  read  status
register
```

4.3 NRF24L01 関連レジスタアドレスのマクロ定義

```
#define CONFIG        0x00   //'Config' register address
#define EN_AA         0x01  //'Enable Auto Acknowledgment' register address
#define EN_RXADDR     0x02  //'Enabled RX addresses' register address
#define SETUP_AW      0x03  //'Setup address width' register address
#define SETUP_RETR    0x04  //'Setup Auto. Retrans' register address
#define RF_CH         0x05  //'RF channel' register address
#define RF_SETUP      0x06  //'RF setup' register address
#define STATUS        0x07  //'Status' register address
#define OBSERVE_TX    0x08  //'Observe TX' register address
#define CD            0x09  //'Carrier Detect' register address
#define RX_ADDR_P0    0x0A    //'RX address pipe0' register address
#define RX_ADDR_P1    0x0B  //'RX address pipe1' register address
#define RX_ADDR_P2    0x0C  //'RX address pipe2' register address
#define RX_ADDR_P3    0x0D    //'RX address pipe3' register address
#define RX_ADDR_P4    0x0E  //'RX address pipe4' register address
#define RX_ADDR_P5    0x0F    //'RX address pipe5' register address
#define TX_ADDR       0x10  //'TX address' register address
#define RX_PW_P0      0x11  //'RX payload width、 pipe0' register address
#define RX_PW_P1      0x12  //'RX payload width、 pipe1' register address
#define RX_PW_P2      0x13  //'RX payload width、 pipe2' register address
#define RX_PW_P3      0x14  //'RX payload width、 pipe3' register address
#define RX_PW_P4      0x15  //'RX payload width、 pipe4' register address
#define RX_PW_P5      0x16  //'RX payload width、 pipe5' register address
```

```
#define FIFO_STATUS       0x17  //'FIFO Status Register' register address
```

5、実際通信プロセスオシロスコープ図

NRF24L01 プログラミングはコマンド（WRITE_REG READ_REG など）、制御ライン CE、CSN と割り込み信号 IRQ で一緒に完成する。

送信ノードについて、ACK と IRQ 機能を有効する場合、通信成功後（送信ノードが受信ノードからフィードバックの ACK 信号を受信）、IRQ ラインは低レベルとなる。

受信ノードについて、ACK と IRQ 機能を有効する場合、通信成功後（Enhanced ShockBurst プロトコルが有効データ幅のデータを受信と判断する）、IRQ ラインは低レベルとなる。

上記の状況に基づき、オシロスコープで下記のグラフを例とする：

1）送信ノード CE と IRQ 信号

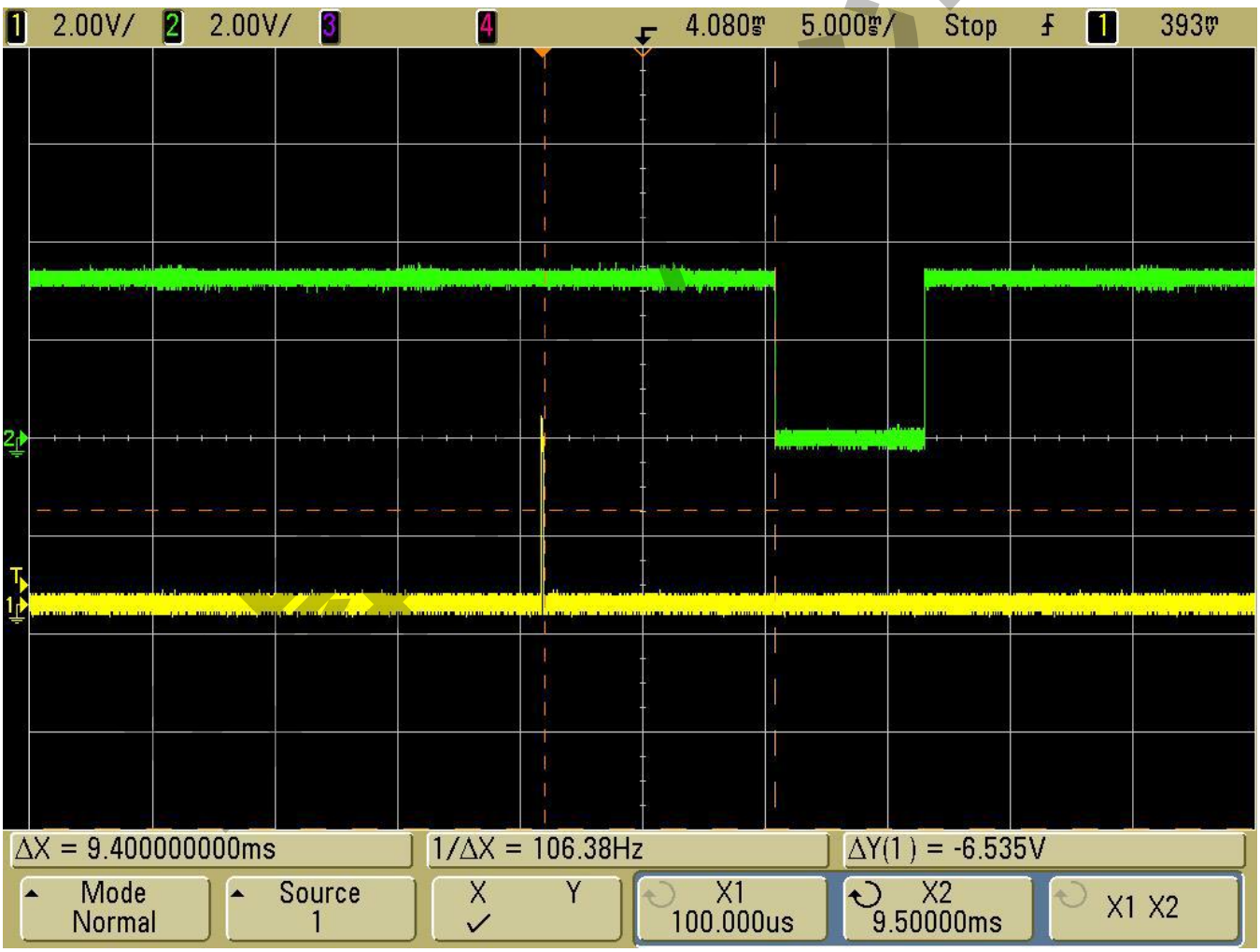


FIG5.1

黄色信号は CE、緑信号は IRQ、ノードを送信ノードに設定後、送信データは SPI_Write_Buf(WRITE_REG + RX_ADDR_P0、 TX_ADDRESS、 TX_ADR_WIDTH)関数を介し、FIFO バッファへ送信する。CE は高レベル、10us を超えると、バッファのデータをワイヤレスで送信する。

IQR のすべての機能を有効する（TX_DS、RX_DS、MAX_RT）と、送信ノードが受信ノードからフィードバック ACK 信号または最大送信回数を達する場合、IRQ は低レベルとなり、CONFIG の関連フラグ（）は 1 と設定

される。フラグクリアする（ CONFIG フラグ＝1）後、IRQ は高レベルとなる。

CE は高レベルになり、10ms 後 IRQ は低レベルに変化する。IRQ は最大送信回数に達し（MAX_RT=1)、この状況となる原因は下記の通り：

- 受信ノードと送信ノードの構成が一致しない（送受信の周波数、送受信のバイト範囲）
- 受信ノードがない（設定しない、通電しない）

2）SCK と IRQ 信号（送信成功）

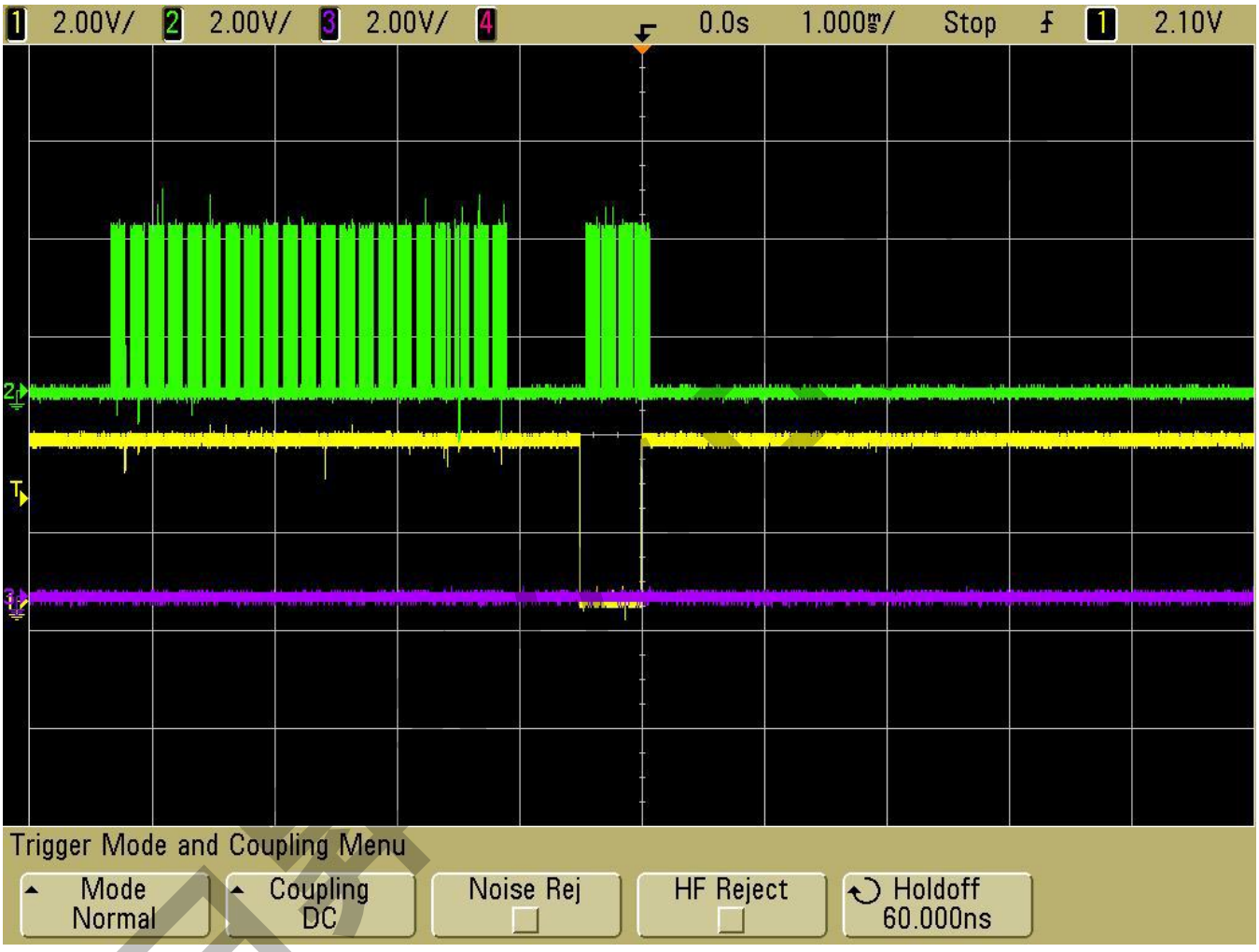


Fig5.2

緑信号は SCK、黄色信号 IRQ。第一配列の緑信号はノードの設定プロセス。MOSI 信号（Fig5.2 で表示しない）は SCK の立ち下がりエッジで 24L01 ノードへ送信する。（1 つのレジスタを設定するには 2 組の SCK 信号が必要、N バイトの BUFFER 書き込むには、N+1 組の SCK 信号が必要とする。）

信号設定完了後、CE（Fig5.2 で表示しない）をハイと設定、24L01 からデータを受送信する。受送信完了（最大送信回数到達）後、IRQ をローに設定。SCM はステータスにより対応動作する。

第二配列の緑信号は IRQ が低レベルの場合、SCM は 24L01 の処理プロセス。FIFO 読み取り（受信ノード)、FIFO 書き込み（送信ノード)、24L01　Reset（最大送信回数)。

Fig5.2 から見て、第一配列 SCK の最後の信号から IRQ が低レベルとなるまで約 1ms（ Fig5.1 は 12ms)、通信成功と表示する（IRQ 変化は MAX_RT と関連しないと表明する)。

3）SCK と IRQ 信号（送信失敗）

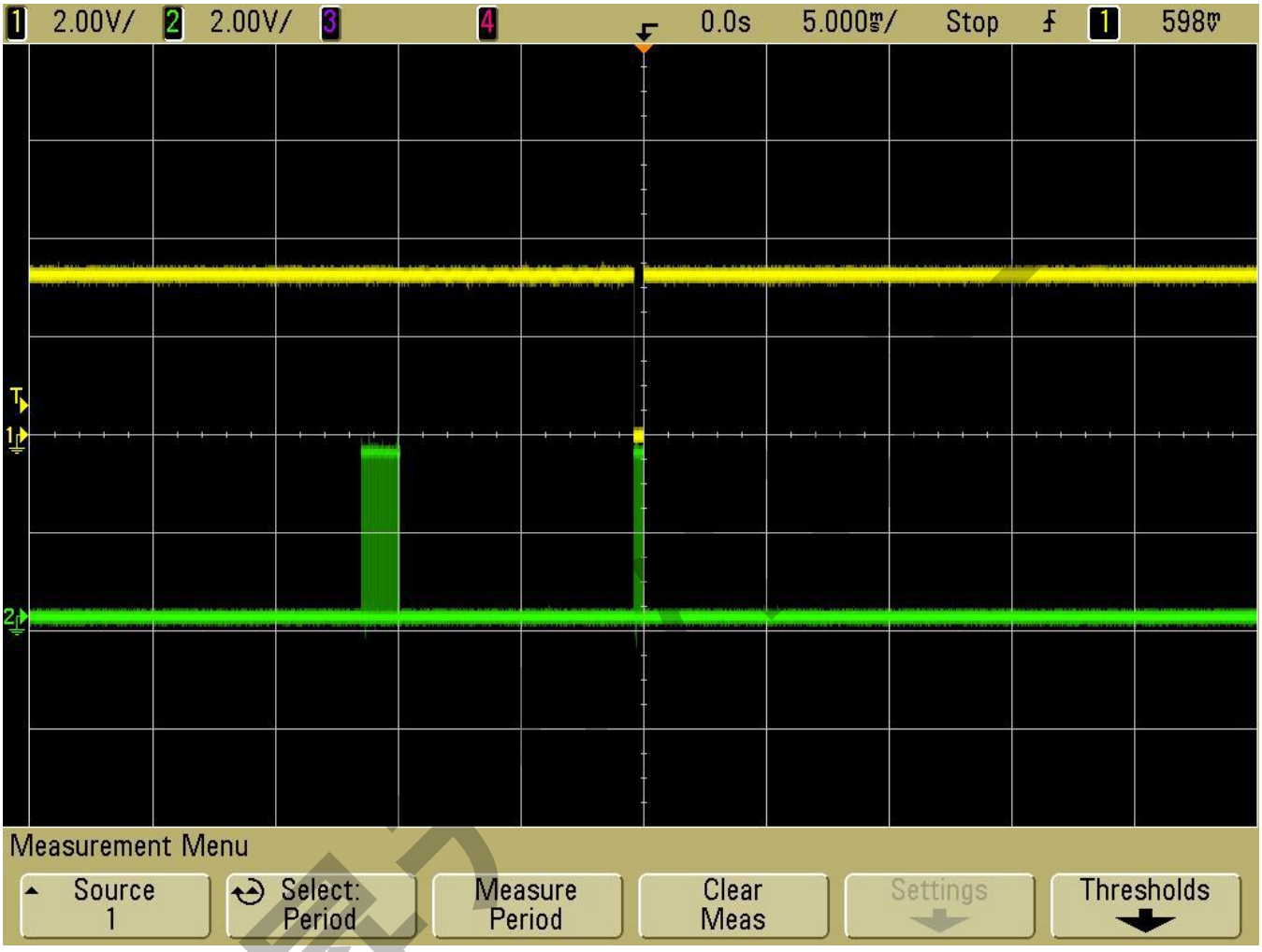


Fig5.3

Fig5.3 は Fig5.2 と類似、第一配列の SCK の最後の信号から IRQ が低レベルとなるまで約 10ms、通信失敗と表示する（IRQ 変化は最大送信回数到達）。

4）SCK、IRQ、CE 信号

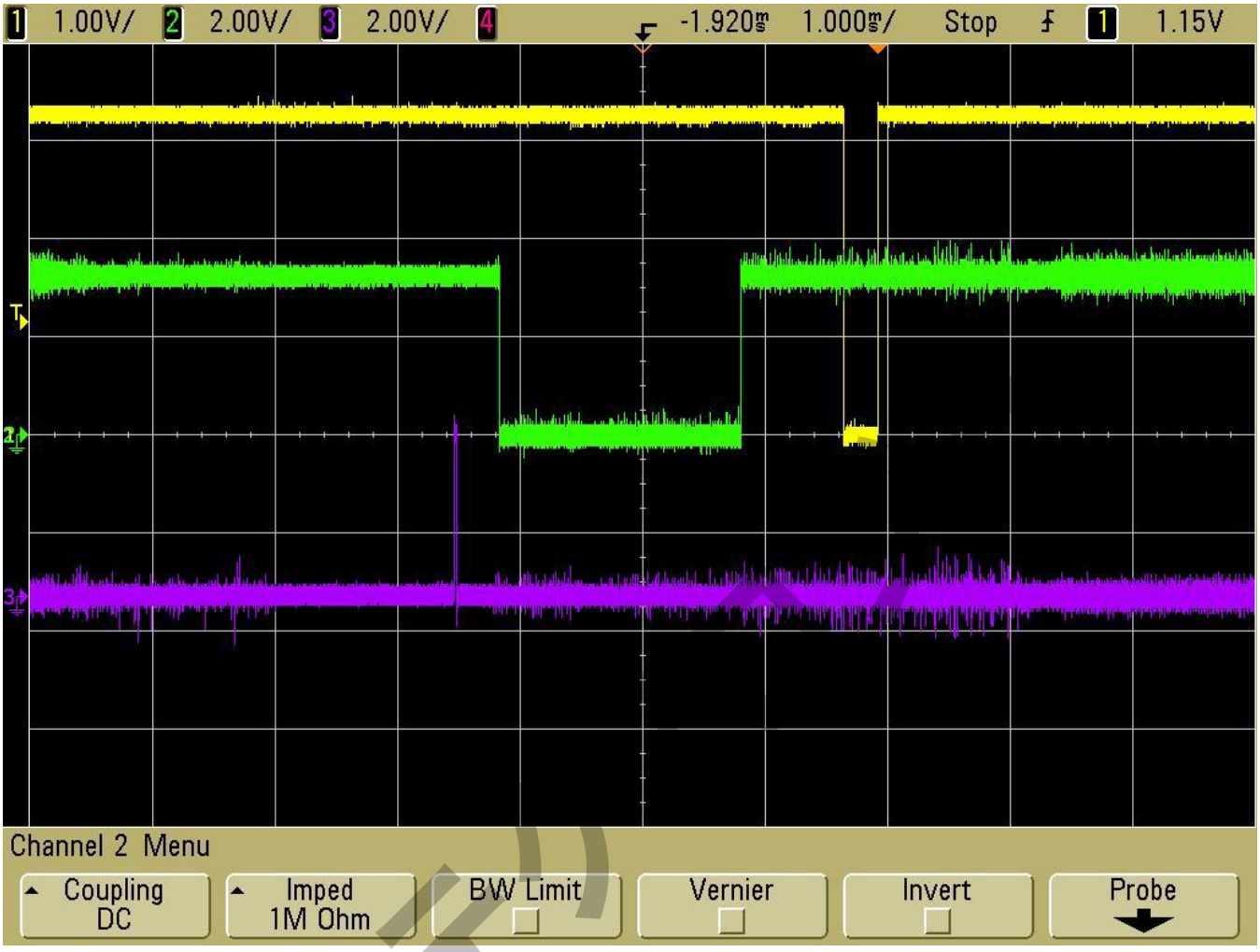


Fig5.4

Fig5.4 パープル信号は送信ノード CE、緑信号は受信ノード IRQ、黄色信号は送信ノード IRQ。

送信ノード設定完了後、CE を高レベルとなり、送信ノードの FIFO のデータを送信；受信ノードデータ受信後、フィードバックし、受信 IRQ 低レベルを設定し（パープル信号と緑信号の時間間隔で送信状況を判断できる）；受信ノード自動的に ACK 信号を送信（ACK 有効する）、送信ノードは ACK 受信後、IRQ を低レベルとする、送信成功と表示。

通信環境により、受送信ノードの IRQ の位相が変わる（オシロスコープで黄色信号と緑信号の間隔）。この状況は通信環境により、受信ノードの ACK 信号の送信時間（失敗、再送）である。