Índice

ANEXO A. EL PIC18F4520 EN LA PLACA OPEN18F4520 ____________ 3
  A.1. Datasheet del PIC18F4520 ................................................................. 3
  A.2. Placa Open18f4520 ........................................................................... 25

ANEXO B. COMUNICACIÓN SPI ENTRE DOS PIC18 ____________ 29
  B.1. Funciones para la comunicación SPI ..................................................... 29
  B.2. Códigos para la comunicación entre dos PIC18 ..................................... 31
    B.2.1. Código para el transmisor ................................................................. 31
    B.2.2. Código para el receptor .................................................................... 33

ANEXO C. CÓDIGOS ANTECEDENTES PARA EL NRF24L01 ______ 35
  C.1. Código de WaveshareElectronics ......................................................... 35
  C.2. Código para AVR de Google Code ...................................................... 43

ANEXO D. INFORMACIÓN DEL NRF24L01 ____________ 49
  D.1. Esquemático y layout del módulo NRF24L01 RF Board ...................... 49
  D.2. Datasheet del nRF24L01 ................................................................... 51
ANEXO A. EL PIC18F4520 en la placa Open18f4520

A.1. Datasheet del PIC18F4520

<table>
<thead>
<tr>
<th>TABLE 1-1: DEVICE FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
</tr>
<tr>
<td>Operating Frequency</td>
</tr>
<tr>
<td>Program Memory (Bytes)</td>
</tr>
<tr>
<td>Program Memory (Instructions)</td>
</tr>
<tr>
<td>Data Memory (Bytes)</td>
</tr>
<tr>
<td>Data EEPROM Memory (Bytes)</td>
</tr>
<tr>
<td>Interrupt Sources</td>
</tr>
<tr>
<td>Timers</td>
</tr>
<tr>
<td>Capture/Compare/PWM Modules</td>
</tr>
<tr>
<td>Enhanced Capture/Compare/PWM Modules</td>
</tr>
<tr>
<td>Serial Communications</td>
</tr>
<tr>
<td>Parallel Communications (PSP)</td>
</tr>
<tr>
<td>10-Bit Analog-to-Digital Module</td>
</tr>
<tr>
<td>Resets (and Delays)</td>
</tr>
<tr>
<td>Programmable High/Low-Voltage Detect</td>
</tr>
<tr>
<td>Programmable Brown-out Reset</td>
</tr>
<tr>
<td>Instruction Set</td>
</tr>
<tr>
<td>Packages</td>
</tr>
</tbody>
</table>
28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology

Power Management Features:
- Run: CPU on, Peripherals on
- Idle: CPU off, Peripherals on
- Sleep: CPU off, Peripherals off
- Ultra Low 50μA Input Leakage
- Run mode Currents Down to 11 μA Typical
- Idle mode Currents Down to 2.5 μA Typical
- Sleep mode Current Down to 100 nA Typical
- Timer1 Oscillator: 900 nA, 32 kHz, 2V
- Watchdog Timer: 1.4 μA, 2V Typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:
- Four Crystal modes, up to 40 MHz
- 4x Phase Lock Loop (PLL) – Available for Crystal and Internal Oscillators
- Two External RC modes, up to 4 MHz
- Two External Clock modes, up to 40 MHz
- Internal Oscillator Block:
  - Fast wake from Sleep and Idle, 1 μs typical
  - 8-use-selectable frequencies, from 31 kHz to 8 MHz
  - Provides a complete range of clock speeds from 31 kHz to 32 MHz when used with PLL
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if peripheral clock stops

Peripheral Highlights:
- High-Current Sink/Source 25 mA/25 mA
- Three Programmable External Interrupts
- Four Input Change Interrupts
- Up to 2 Capture/Compare/PWM (CCP) modules, one with Auto-Shutdown (28-pin devices)
- Enhanced Capture/Compare/PWM (ECCP) module (40/44-pin devices only):
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart

Peripheral Highlights (Continued):
- Master Synchronous Serial Port (MSSP) module
- Supporting 3-Wire SPI (all 4 modes) and I²C™ Master and Slave modes
- Enhanced Addressable USART module:
  - Supports RS-485, RS-232 and LIN/J2602
  - RS-232 operation using internal oscillator
    block (no external crystal required)
  - Auto-wake-up on Start bit
  - Auto-Baud Detect
- 10-Bit, up to 13-Channel Analog-to-Digital (A/D) Converter module:
  - Auto-acquisition capability
  - Conversion available during Sleep
  - Dual Analog Comparators with Input Multiplexing
  - Programmable 16-Level High/Low-Voltage
    Detection (H/LVD) module:
    - Supports interrupt on High/Low-Voltage Detection

Special Microcontroller Features:
- C Compiler Optimized Architecture:
  - Optional extended instruction set designed to
    optimize re-entrant code
- 100,000 Erase/Write Cycle Enhanced Flash
  Program Memory Typical
- 1,000,000 Erase/Write Cycle Data EEPROM
  Memory Typical
- Flash/Data EEPROM Retention: 100 Years Typical
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 4 ms to 131s
- Single-Supply 5V In-Circuit Serial
  Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) via Two Pins
- Wide Operating Voltage Range: 2.0V to 5.5V
- Programmable Brown-out Reset (BOR) with
  Software Enable Option

<table>
<thead>
<tr>
<th>Device</th>
<th>Program Memory</th>
<th>Data Memory</th>
<th>I/O</th>
<th>10-Bit A/D (ch)</th>
<th>CCP/ ECCP (PWM)</th>
<th>MSSP</th>
<th>SPI</th>
<th>Master I²C™</th>
<th>EUSART</th>
<th>Comp.</th>
<th>Timers 8/16-Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC18F2420</td>
<td>16K</td>
<td>8192</td>
<td>768</td>
<td>256</td>
<td>25</td>
<td>10</td>
<td>Y</td>
<td>Y</td>
<td>1</td>
<td>2</td>
<td>1/3</td>
</tr>
<tr>
<td>PIC18F2520</td>
<td>32K</td>
<td>16384</td>
<td>1536</td>
<td>256</td>
<td>25</td>
<td>10</td>
<td>Y</td>
<td>Y</td>
<td>1</td>
<td>2</td>
<td>1/3</td>
</tr>
<tr>
<td>PIC18F4420</td>
<td>16K</td>
<td>8192</td>
<td>768</td>
<td>256</td>
<td>36</td>
<td>13</td>
<td>1/1</td>
<td>Y</td>
<td>Y</td>
<td>1</td>
<td>2/1</td>
</tr>
<tr>
<td>PIC18F4520</td>
<td>32K</td>
<td>16384</td>
<td>1536</td>
<td>256</td>
<td>36</td>
<td>13</td>
<td>1/1</td>
<td>Y</td>
<td>Y</td>
<td>1</td>
<td>2/3</td>
</tr>
</tbody>
</table>
PIC18F2420/2520/4420/4520

FIGURE 1-2: PIC18F4420/4520 (40/44-PIN) BLOCK DIAGRAM

Note 1: CCP2 is multiplexed with RC1 when Configuration bit CCP2MX is set, or RB3 when CCP2MX is not set.
2: REI is only available when MCLR functionality is disabled.
3: OSC1/CLK1 and OSC2/CLK0 are only available in select oscillator modes and when these pins are not being used as digital I/O.
   Refer to Section 2.6 “Oscillator Configurations” for additional information.
2.0 OSCILLATOR CONFIGURATIONS

2.1 Oscillator Types

PIC18F2420/2520/4420/4520 devices can be operated in ten different oscillator modes. The user can program the Configuration bits, FOSC<3:0>, in Configuration Register 1H to select one of these ten modes:

1. LP Low-Power Crystal
2. XT Crystal/Resonator
3. HS High-Speed Crystal/Resonator
4. HSPLL High-Speed Crystal/Resonator with PLL Enabled
5. RC External Resistor/Capacitor with Fosc/4 Output on RA6
6. RCIO External Resistor/Capacitor with I/O on RA6
7. INTIO1 Internal Oscillator with Fosc/4 Output on RA6 and I/O on RA7
8. INTIO2 Internal Oscillator with I/O on RA6 and RA7
9. EC External Clock with Fosc/4 Output
10. ECIO External Clock with I/O on RA6

2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS or HSPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-1 shows the pin connections.

The oscillator design requires the use of a parallel cut crystal.

Note: Use of a series cut crystal may give a frequency out of the crystal manufacturer's specifications.

![CRYSTAL/CERAMIC RESONATOR OPERATION (XT, LP, HS OR HSPLL CONFIGURATION)]

Note 1: See Table 2-1 and Table 2-2 for initial values of C1 and C2.
2: A series resistor (Rs) may be required for AT strip cut crystals.
3: RF varies with the oscillator mode chosen.

### TABLE 2-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

<table>
<thead>
<tr>
<th>Mode</th>
<th>Freq</th>
<th>OSC1</th>
<th>OSC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>XT</td>
<td>3.56 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
</tr>
<tr>
<td></td>
<td>4.19 MHz</td>
<td>15 pF</td>
<td>15 pF</td>
</tr>
<tr>
<td></td>
<td>4 MHz</td>
<td>30 pF</td>
<td>30 pF</td>
</tr>
<tr>
<td></td>
<td>4 MHz</td>
<td>50 pF</td>
<td>50 pF</td>
</tr>
</tbody>
</table>

Capacitor values are for design guidance only.

Different capacitor values may be required to produce acceptable oscillator operation. The user should test the performance of the oscillator over the expected VDD and temperature range for the application.

See the notes following Table 2-2 for additional information.

Note: When using resonators with frequencies above 3.5 MHz, the use of HS mode, rather than XT mode, is recommended. HS mode may be used at any VDD for which the controller is rated. If HS is selected, it is possible that the gain of the oscillator will overcome the resonator. Therefore, a series resistor should be placed between the OSC2 pin and the resonator. As a good starting point, the recommended value of Rs is 330Ω.
5.0 MEMORY ORGANIZATION

There are three types of memory in PIC18 enhanced microcontroller devices:

- Program Memory
- Data RAM
- Data EEPROM

As Harvard architecture devices, the data and program memories use separate busses; this allows for concurrent access of the two memory spaces. The data EEPROM, for practical purposes, can be regarded as a peripheral device, since it is addressed and accessed through a set of control registers.

Additional detailed information on the operation of the Flash program memory is provided in Section 6.0 “Flash Program Memory”. Data EEPROM is discussed separately in Section 7.0 “Data EEPROM Memory”.

5.1 Program Memory Organization

PIC18 microcontrollers implement a 21-bit program counter, which is capable of addressing a 2-Mbyte program memory space. Accessing a location between the upper boundary of the physically implemented memory and the 2-Mbyte address will return all ‘0’ s (a NOP instruction).

The PIC18F2420 and PIC18F4420 each have 16 Kbytes of Flash memory and can store up to 8,192 single-word instructions. The PIC18F2520 and PIC18F4520 each have 32 Kbytes of Flash memory and can store up to 16,384 single-word instructions.

PIC18 devices have two interrupt vectors. The Reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

The program memory map for PIC18F2420/2520/4420/4520 devices is shown in Figure 5-1.

**FIGURE 5-1: PROGRAM MEMORY MAP AND STACK FOR PIC18F2420/2520/4420/4520 DEVICES**
FIGURE 5-6: DATA MEMORY MAP FOR PIC18F2520/4520 DEVICES

<table>
<thead>
<tr>
<th>BSR&lt;3:0&gt;</th>
<th>Bank 0</th>
<th>Bank 1</th>
<th>Bank 2</th>
<th>Bank 3</th>
<th>Bank 4</th>
<th>Bank 5</th>
<th>Bank 6</th>
<th>Bank 7</th>
<th>Bank 8</th>
<th>Bank 9</th>
<th>Bank 10</th>
<th>Bank 11</th>
<th>Bank 12</th>
<th>Bank 13</th>
<th>Bank 14</th>
<th>Bank 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>= 0000</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 0001</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 0010</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 0011</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 0100</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 0101</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 0110</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 0111</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 1000</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 1001</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 1010</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 1011</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 1100</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 1101</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 1110</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
<tr>
<td>= 1111</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>00h</td>
<td>FFh</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
<td>00h</td>
<td>FFh</td>
</tr>
</tbody>
</table>

Data Memory Map

- **Bank 0**: 00h-00h
- **Bank 1**: 00h-FFh
- **Bank 2**: 00h-FFh
- **Bank 3**: 00h-FFh
- **Bank 4**: 00h-FFh
- **Bank 5**: 00h-FFh
- **Bank 6**: 00h-FFh
- **Bank 7**: 00h-FFh
- **Bank 8**: 00h-FFh
- **Bank 9**: 00h-FFh
- **Bank 10**: 00h-FFh
- **Bank 11**: 00h-FFh
- **Bank 12**: 00h-FFh
- **Bank 13**: 00h-FFh
- **Bank 14**: 00h-FFh
- **Bank 15**: 00h-FFh

### Access Bank
- **Access RAM Low**: 00h-07h
- **Access RAM High (SFRs)**: 00h-FFh

### Data Access
- **Bank 0**: 00h-07h
- **Bank 1**: 08h-0Fh
- **Bank 2**: 10h-1Fh
- **Bank 3**: 20h-2Fh
- **Bank 4**: 30h-3Fh
- **Bank 5**: 40h-4Fh
- **Bank 6**: 50h-5Fh
- **Bank 7**: 60h-6Fh
- **Bank 8**: 70h-7Fh
- **Bank 9**: 80h-8Fh
- **Bank 10**: 90h-9Fh
- **Bank 11**: A0h-AFh
- **Bank 12**: B0h-BFh
- **Bank 13**: C0h-CFh
- **Bank 14**: D0h-DFh
- **Bank 15**: EFh-FFh

### Usage
- **Access RAM**: 00h-07h
- **Special Function Registers (SFRs)**: 08h-FFh

When `a` = 0:
The BSR is ignored and the Access Bank is used.
The first 128 bytes are general purpose RAM (from Bank 0).
The second 128 bytes are Special Function Registers (from Bank 15).

When `a` = 1:
The BSR specifies the Bank used by the instruction.
9.0 INTERRUPTS

The PIC18F2420/2520/4420/4520 devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high-priority level or a low-priority level. The high-priority interrupt vector is at 0008h and the low-priority interrupt vector is at 0018h. High-priority interrupt events will interrupt any low-priority interrupts that may be in progress.

There are ten registers which are used to control interrupt operation. These registers are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2
- PIE1, PIE2
- IPR1, IPR2

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

In general, interrupt sources have three bits to control their operation. They are:

- **Flag bit** to indicate that an interrupt event occurred
- **Enable bit** that allows program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** to select high priority or low priority

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIIEH bit (INTCON<7>) enables all interrupts that have the priority bit set (high priority). Setting the GIIEL bit (INTCON<8>) enables all interrupts that have the priority bit cleared (low priority). When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 0008h or 0018h, depending on the priority bit setting. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit, which enables/disables all peripheral interrupt sources. INTCON<7> is the GIIE bit, which enables/disables all interrupt sources. All interrupts branch to address 0008h in Compatibility mode.

When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIIEH or GIIEL bit. High-priority interrupt sources can interrupt a low-priority interrupt. Low-priority interrupts are not processed when high-priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

The "return from interrupt" instruction, RETFIE, exits the interrupt routine and sets the GIE bit (GIIEH or GIIEL if priority levels are used), which re-enables interrupts.

For external interrupt events, such as the INTX pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one or two-cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding enable bit or the GIE bit.

**Note:** Do not use the MOVFF instruction to modify any of the interrupt control registers while any interrupt is enabled. Doing so may cause erratic microcontroller behavior.
9.2 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Request Flag registers (PIR1 and PIR2).

**Note 1:** Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the Global Interrupt Enable bit, GIE (INTCON<7>).

**Note 2:** User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt and after servicing that interrupt.

### REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPRIF(1)</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSPIF</td>
<td>CCP1IF</td>
<td>TMR2IF</td>
<td>TMR1IF</td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **R** = Readable bit
- **W** = Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **-n** = Value at POR
- **‘1’** = Bit is set
- **‘0’** = Bit is cleared
- **x** = Bit is unknown

- **bit 7** PSPRIF: Parallel Slave Port Read/Write Interrupt Flag bit(1)
  - 1 = A read or a write operation has taken place (must be cleared in software)
  - 0 = No read or write has occurred
- **bit 6** ADIF: A/D Converter Interrupt Flag bit
  - 1 = An A/D conversion completed (must be cleared in software)
  - 0 = The A/D conversion is not complete
- **bit 5** RCIF: EUSART Receive Interrupt Flag bit
  - 1 = The EUSART receive buffer, RCREG, is full (cleared when RCREG is read)
  - 0 = The EUSART receive buffer is empty
- **bit 4** TXIF: EUSART Transmit Interrupt Flag bit
  - 1 = The EUSART transmit buffer, TXREG, is empty (cleared when TXREG is written)
  - 0 = The EUSART transmit buffer is full
- **bit 3** SSPRIF: Master Synchronous Serial Port Interrupt Flag bit
  - 1 = The transmission/reception is complete (must be cleared in software)
  - 0 = Waiting to transmit/receive
- **bit 2** CCP1IF: CCP1 Interrupt Flag bit
  - **Capture mode:**
    - 1 = A TMR1 register capture occurred (must be cleared in software)
    - 0 = No TMR1 register capture occurred
  - **Compare mode:**
    - 1 = A TMR1 register compare match occurred (must be cleared in software)
    - 0 = No TMR1 register compare match occurred
  - **PWM mode:**
    - Unused in this mode.
- **bit 1** TMR2IF: TMR2 to PR2 Match Interrupt Flag bit
  - 1 = TMR2 to PR2 match occurred (must be cleared in software)
  - 0 = No TMR2 to PR2 match occurred
- **bit 0** TMR1IF: TMR1 Overflow Interrupt Flag bit
  - 1 = TMR1 register overflowed (must be cleared in software)
  - 0 = TMR1 register did not overflow

**Note 1:** This bit is unimplemented on 28-pin devices and will read as ‘0’.
9.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Enable registers (PIE1 and PIE2). When IPEN = 0, the PEIE bit must be set to enable any of these peripheral interrupts.

**REGISTER 9-6: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value at POR</th>
<th>Writable bit</th>
<th>Unimplemented bit, read as 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td><strong>PSPIE</strong> (Parallel Slave Port Read/Write Interrupt Enable bit)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 = Enables the PSP read/write interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Disables the PSP read/write interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>ADIE</strong> (A/D Converter Interrupt Enable bit)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 = Enables the A/D interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Disables the A/D interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>RCIE</strong> (EUSART Receive Interrupt Enable bit)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 = Enables the EUSART receive interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Disables the EUSART receive interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>TXIE</strong> (EUSART Transmit Interrupt Enable bit)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 = Enables the EUSART transmit interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Disables the EUSART transmit interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><strong>SSPIE</strong> (Master Synchronous Serial Port Interrupt Enable bit)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 = Enables the MSSP interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Disables the MSSP interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>CCP1IE</strong> (CCP1 Interrupt Enable bit)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 = Enables the CCP1 interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Disables the CCP1 interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td><strong>TMR2IE</strong> (TMR2 to PR2 Match Interrupt Enable bit)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 = Enables the TMR2 to PR2 match interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Disables the TMR2 to PR2 match interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td><strong>TMR1IE</strong> (TMR1 Overflow Interrupt Enable bit)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 = Enables the TMR1 overflow interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 = Disables the TMR1 overflow interrupt</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as 0
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

**Note:** This bit is unimplemented on 28-pin devices and will read as ‘0'.
9.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are two Peripheral Interrupt Priority registers (IPR1 and IPR2). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

REGISTER 9-8: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

<table>
<thead>
<tr>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
<th>R/W-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSPiP</td>
<td>ADIP</td>
<td>RCIP</td>
<td>TXIP</td>
<td>SSPIP</td>
<td>CCP1IP</td>
<td>TMR2IP</td>
<td>TMR1IP</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- n = Value at POR

- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PSPiP: Parallel Slave Port Read/Write Interrupt Priority bit(^{(1)})</td>
<td>1 = High priority, 0 = Low priority</td>
</tr>
<tr>
<td>6</td>
<td>ADIP: A/D Converter Interrupt Priority bit</td>
<td>1 = High priority, 0 = Low priority</td>
</tr>
<tr>
<td>5</td>
<td>RCIP: EUSART Receive Interrupt Priority bit</td>
<td>1 = High priority, 0 = Low priority</td>
</tr>
<tr>
<td>4</td>
<td>TXIP: EUSART Transmit Interrupt Priority bit</td>
<td>1 = High priority, 0 = Low priority</td>
</tr>
<tr>
<td>3</td>
<td>SSPIP: Master Synchronous Serial Port Interrupt Priority bit</td>
<td>1 = High priority, 0 = Low priority</td>
</tr>
<tr>
<td>2</td>
<td>CCP1IP: CCP1 Interrupt Priority bit</td>
<td>1 = High priority, 0 = Low priority</td>
</tr>
<tr>
<td>1</td>
<td>TMR2IP: TMR2 to PR2 Match Interrupt Priority bit</td>
<td>1 = High priority, 0 = Low priority</td>
</tr>
<tr>
<td>0</td>
<td>TMR1IP: TMR1 Overflow Interrupt Priority bit</td>
<td>1 = High priority, 0 = Low priority</td>
</tr>
</tbody>
</table>

Note 1: This bit is unimplemented on 28-pin devices and will read as ‘0’.
17.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

17.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D Converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I²C)
  - Full Master mode
  - Slave mode (with general address call)

The I²C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

17.2 Control Registers

The MSSP module has three associated registers. These include a status register (SSPSTAT) and two control registers (SSPCON1 and SSPCON2). The use of these registers and their individual configuration bits differ significantly depending on whether the MSSP module is operated in SPI or I²C mode.

Additional details are provided under the individual sections.

17.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

- Serial Data Out (SDO) – RCS/SDO
- Serial Data In (SDI) – RC4/SDI/SDA
- Serial Clock (SCK) – RC3/SCK/SCL

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select (SS) – RA5/SS

Figure 17-1 shows the block diagram of the MSSP module when operating in SPI mode.

---

**FIGURE 17-1: MSSP BLOCK DIAGRAM (SPI MODE)**

- **Read**
  - SSPBUF reg
- **Write**
  - SSPSR reg
- **Shift Clock**
  - Bit 0
- **Control Enable**
  - SS
  - Edge Select
  - Clock Select
  - SPI/CKE
  - SMP/CKE
- **Program (TMR2 Output)**
  - 2
  - 4
  - 16
  - 64
- **Data to TX/RX in SSPSR**
  - TRIS bit

© 2008 Microchip Technology Inc.
PIC18F2420/2520/4420/4520

17.3.1 REGISTERS
The MSSP module has four registers for SPI mode operation. These are:
• MSSP Control Register 1 (SSPCON1)
• MSSP Status Register (SSPSTAT)
• Serial Receive/Transmit Buffer Register (SSPBUF)
• MSSP Shift Register (SSPSR) – Not directly accessible

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

REGISTER 17-1: SSPSTAT: M SSP STATUS REGISTER (SPI MODE)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMP</td>
<td>CKE[1]</td>
<td>D/Ä</td>
<td>P</td>
<td>S</td>
<td>R/W</td>
<td>UA</td>
<td>BF</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- n = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 7
- **SMP**: Sample bit
  - SPI Master mode:
    - 1 = Input data sampled at end of data output time
    - 0 = Input data sampled at middle of data output time
  - SPI Slave mode:
    - SMP must be cleared when SPI is used in Slave mode.

bit 6
- **CKE**: SPI Clock Select bit[1]
  - 1 = Transmit occurs on transition from active to Idle clock state
  - 0 = Transmit occurs on transition from Idle to active clock state

bit 5
- **D/Ä**: Data/Address bit
  - Used in I²C™ mode only.

bit 4
- **P**: Stop bit
  - Used in I²C™ mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3
- **S**: Start bit
  - Used in I²C™ mode only.

bit 2
- **R/W**: Read/Write Information bit
  - Used in I²C™ mode only.

bit 1
- **UA**: Update Address bit
  - Used in I²C™ mode only.

bit 0
- **BF**: Buffer Full Status bit (Receive mode only)
  - 1 = Receive complete, SSPBUF is full
  - 0 = Receive not complete, SSPBUF is empty

**Note 1**: Polarity of clock state is set by the CKP bit (SSPCON1<4>).
Comunicación por RF entre microcontroladores PIC18 mediante el módulo NRF24L01

PIC18F2420/2520/4420/4520

REGISTER 17-2: MSSP CONTROL REGISTER 1 (SPI MODE)

<table>
<thead>
<tr>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPOV</td>
<td>SSPOV</td>
<td>SSPOV</td>
<td>SSPOV</td>
<td>SSPOV</td>
<td>SSPOV</td>
</tr>
<tr>
<td>CKP</td>
<td>SMP0</td>
<td>SMP0</td>
<td>SMP0</td>
<td>SMP0</td>
<td>SMP0</td>
<td>SMP0</td>
<td>SMP0</td>
</tr>
</tbody>
</table>

Legend:
R = Readable bit
W = Writable bit
U = Unimplemented bit, read as ‘0’
-n = Value at POR
‘1’ = Bit is set
‘0’ = Bit is cleared
x = Bit is unknown

bit 7   WCOL: Write Collision Detect bit
1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6   SSPOV: Receive Overflow Indicator bit(f)
SPI Slave mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode. The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
0 = No overflow

bit 5   SSPEN: Master Synchronous Serial Port Enable bit(2)
1 = Enables serial port and configures SCK, SDO, SDI and SS as serial port pins
0 = Disables serial port and configures these pins as I/O port pins

bit 4   CKP: Clock Polarity Select bit
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

bit 3-0   SSPM<3:0>: Master Synchronous Serial Port Mode Select bits(3)
0101 = SPI Slave mode, clock = SCK pin; SS pin control disabled; SS can be used as I/O pin
0100 = SPI Slave mode, clock = SCK pin; SS pin control enabled
0011 = SPI Master mode, clock = TMR2 output/2
0010 = SPI Master mode, clock = Fosc/64
0001 = SPI Master mode, clock = Fosc/16
0000 = SPI Master mode, clock = Fosc/4

Note 1: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.
2: When enabled, these pins must be properly configured as input or output.
3: Bit combinations not specifically listed here are either reserved or implemented in I²C™ mode only.
PIC18F2420/2520/4420/4520

17.3.2 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0> and SSPSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

The MSSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR until the received data is ready. Once the 8 bits of data have been received, that byte is moved to the SSPBUF register. Then, the Buffer Full detect bit, BF (SSPSTAT<0>) and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the write collision detect bit, WCOL (SSPCON1<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. The Buffer Full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. The SSPBUF must be read and/or written. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur. Example 17-1 shows the loading of the SSPBUF (SSPSR) for data transmission. The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP Status register (SSPSTAT) indicates the various status conditions.

**Note:** The SSPBUF register cannot be used with read-modify-write instructions such as BCF, BTFSC and COMF, etc.

---

**EXAMPLE 17-1: LOADING THE SSPBUF (SSPSR) REGISTER**

<table>
<thead>
<tr>
<th>LOOP</th>
<th>BTFS</th>
<th>SSPSTAT, BF ; Has data been received (transmit complete)?</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRA</td>
<td>LOOP</td>
<td>; No</td>
</tr>
<tr>
<td>MOVF</td>
<td>SSPBUF, W ; WRBU reg = contents of SSPBUF</td>
<td></td>
</tr>
<tr>
<td>MOVWF</td>
<td>RXDATA ; Save in user RAM, if data is meaningful</td>
<td></td>
</tr>
<tr>
<td>MOVF</td>
<td>TXDATA, W ; W reg = contents of TXDATA</td>
<td></td>
</tr>
<tr>
<td>MOVWF</td>
<td>SSPBUF ; New data to xmit</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** To avoid lost data in Master mode, a read of the SSPBUF must be performed to clear the Buffer Full (BF) detect bit (SSPSTAT<0>) between each transmission.
17.3.3 ENABLING SPI I/O

To enable the serial port, MSSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize the SSPCON registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and SS pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> bit cleared
- SCK (Master mode) must have TRISC<3> bit cleared
- SCK (Slave mode) must have TRISC<3> bit set
- SS must have TRISA<5> bit set

Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

17.3.4 TYPICAL CONNECTION

Figure 17-2 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data – Slave sends dummy data
- Master sends data – Slave sends data
- Master sends dummy data – Slave sends data

**FIGURE 17-2: SPI MASTER/SLAVE CONNECTION**
17.3.5 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 17-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a “Line Activity Monitor” mode.

The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This, then, would give waveforms for SPI communication as shown in Figure 17-3, Figure 17-5 and Figure 17-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user-programmable to be one of the following:

- FOSC/4 (or Tcy)
- FOSC/16 (or 4 • Tcy)
- FOSC/64 (or 16 • Tcy)
- Timer2 output/2

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 17-3 shows the waveforms for Master mode. When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPBUF is loaded with the received data is shown.

**FIGURE 17-3:** SPI MODE WAVEFORM (MASTER MODE)
17.3.6 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag bit is set.

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit (SSPCON1<4>).

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. When a byte is received, the device will wake-up from Sleep.

17.3.7 SLAVE SELECT SYNCHRONIZATION

The SS pin allows a Synchronous Slave mode. The SPI must be in Slave mode with SS pin control enabled (SSPCON1<3:0> = 04h). The pin must not be driven low for the SS pin to function as an input. The data latch must be high. When the SS pin is low, transmission and reception are enabled and the SDO pin is driven. When the SS pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

Note 1: When the SPI is in Slave mode with SS pin control enabled (SSPCON<3:0> = 0100), the SPI module will reset if the SS pin is set to VDD.

2: If the SPI is used in Slave mode with CKE set, then the SS pin control must be enabled.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the SS pin to a high level or clearing the SSPEN bit.

To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function) since it cannot create a bus conflict.
PIC18F2420/2520/4420/4520

FIGURE 17-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

FIGURE 17-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)
17.3.8 OPERATION IN POWER-MANAGED MODES

In SPI Master mode, module clocks may be operating at a different speed than when in full-power mode; in the case of Sleep mode, all clocks are halted.

In most Idle modes, a clock is provided to the peripherals. That clock should be from the primary clock source, the secondary clock (Timer1 oscillator at 32.768 kHz) or the INTOSC source. See Section 2.7 “Clock Sources and Oscillator Switching” for additional information.

In most cases, the speed that the master clocks SPI data is not important; however, this should be evaluated for each system.

If MSSP interrupts are enabled, they can wake the controller from Sleep mode, or one of the Idle modes, when the master completes sending data. If an exit from Sleep or Idle mode is not desired, MSSP interrupts should be disabled.

If the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the devices wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in any power-managed mode and data to be shifted into the SPI Transmit/Receive Shift register. When all 8 bits have been received, the MSSP interrupt flag bit will be set, and if enabled, will wake the device.

17.3.9 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

17.3.10 BUS MODE COMPATIBILITY

Table 17-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

<table>
<thead>
<tr>
<th>Standard SPI Mode Terminology</th>
<th>Control Bits State</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CKP</td>
</tr>
<tr>
<td>0, 0</td>
<td>0</td>
</tr>
<tr>
<td>0, 1</td>
<td>0</td>
</tr>
<tr>
<td>1, 0</td>
<td>1</td>
</tr>
<tr>
<td>1, 1</td>
<td>1</td>
</tr>
</tbody>
</table>

There is also an SMP bit which controls when the data is sampled.

### TABLE 17-1: SPI BUS MODES

#### TABLE 17-2: REGISTERS ASSOCIATED WITH SPI OPERATION

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>Reset Values on page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTCON</td>
<td>GIE/GIEH</td>
<td>PEIE/PEIEL</td>
<td>TMR0IE</td>
<td>INTOIE</td>
<td>RBIE</td>
<td>TMR0IF</td>
<td>INTOIF</td>
<td>RBIF</td>
<td>49</td>
</tr>
<tr>
<td>PIR1</td>
<td>PSPIF</td>
<td>ADIF</td>
<td>RCIF</td>
<td>TXIF</td>
<td>SSP1F</td>
<td>CCP1F</td>
<td>TRM2IF</td>
<td>TRM1IF</td>
<td>52</td>
</tr>
<tr>
<td>PIE1</td>
<td>PSPIE</td>
<td>ADIE</td>
<td>RCIE</td>
<td>TXIE</td>
<td>SSP1E</td>
<td>CCP1E</td>
<td>TRM2IE</td>
<td>TRM1IE</td>
<td>52</td>
</tr>
<tr>
<td>IPR1</td>
<td>PSPIP</td>
<td>ADIP</td>
<td>RCIP</td>
<td>TXIP</td>
<td>SSP1P</td>
<td>CCP1P</td>
<td>TRM2IP</td>
<td>TRM1IP</td>
<td>52</td>
</tr>
<tr>
<td>TRISA</td>
<td>TRISA7</td>
<td>TRISA6</td>
<td>PORTA Data Direction Register</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRISC</td>
<td>PORTC Data Direction Register</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPBUF</td>
<td>MSSP Receive Buffer/Transmit Register</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSPCON1</td>
<td>WCOL</td>
<td>SSPOV</td>
<td>SSPEN</td>
<td>CKP</td>
<td>SSPM3</td>
<td>SSPM2</td>
<td>SSPM1</td>
<td>SSPM0</td>
<td>50</td>
</tr>
<tr>
<td>SSPSTAT</td>
<td>SMP</td>
<td>CKE</td>
<td>D/Ã-</td>
<td>P</td>
<td>S</td>
<td>RW</td>
<td>UA</td>
<td>BF</td>
<td>50</td>
</tr>
</tbody>
</table>

Legend: Shaded cells are not used by the MSSP in SPI mode.

Note 1: These bits are unimplemented in 28-pin devices; always maintain these bits clear.

Note 2: PORTA<7:6> and their direction bits are individually configured as port pins based on various primary oscillator modes. When disabled, these bits read as ‘0’.

© 2008 Microchip Technology Inc.
19.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has 10 inputs for the 28-pin devices and 13 for the 40/44-pin devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

The module has five registers:
- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

REGISTER 19-1: ADCON0: A/D CONTROL REGISTER 0

<table>
<thead>
<tr>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHS3</td>
<td>CHS2</td>
<td>CHS1</td>
<td>CHS0</td>
<td>GO/DONE</td>
<td>ADON</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W =Writable bit
- U = Unimplemented bit, read as ‘0’
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 7-6  Unimplemented: Read as ‘0’

bit 5-2  CHS<3:0>: Analog Channel Select bits
0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)\(^{1,2}\)
0110 = Channel 6 (AN6)\(^{1,2}\)
0111 = Channel 7 (AN7)\(^{1,2}\)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)
1101 = Unimplemented\(^2\)
1110 = Unimplemented\(^2\)
1111 = Unimplemented\(^2\)

bit 1  GO/DONE: A/D Conversion Status bit

When ADON = 1:
1 = A/D conversion in progress
0 = A/D Idle

bit 0  ADON: A/D On bit
1 = A/D Converter module is enabled
0 = A/D Converter module is disabled

Note 1: These channels are not implemented on 28-pin devices.

2: Performing a conversion on unimplemented channels will return a floating input measurement.
**PIC18F2420/2520/4420/4520**

**REGISTER 19-2: ADCON1: A/D CONTROL REGISTER 1**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td>VCFG1</td>
<td>VCFG0</td>
<td>PCFG3</td>
<td>PCFG2</td>
<td>PCFG1</td>
<td>PCFG0</td>
</tr>
</tbody>
</table>

Legend:
- **R** = Readable bit
- **W** =Writable bit
- **U** = Unimplemented bit, read as ‘0’
- **n** = Value at POR
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- **x** = Bit is unknown

<table>
<thead>
<tr>
<th>bit 7-6</th>
<th>Unimplemented: Read as ‘0’</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 5</td>
<td><strong>VCFG1</strong>: Voltage Reference Configuration bit (VREF- source)</td>
</tr>
<tr>
<td></td>
<td>1 = VREF- (AN2)</td>
</tr>
<tr>
<td></td>
<td>0 = VSS</td>
</tr>
<tr>
<td>bit 4</td>
<td><strong>VCFG0</strong>: Voltage Reference Configuration bit (VREF+ source)</td>
</tr>
<tr>
<td></td>
<td>1 = VREF+ (AN3)</td>
</tr>
<tr>
<td></td>
<td>0 = VDD</td>
</tr>
<tr>
<td>bit 3-0</td>
<td><strong>PCFG&lt;3:0&gt;: A/D Port Configuration Control bits:</strong>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0000[1]</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>0001</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>0010</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>0011</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>0100</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>0101</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>0110</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>0111[1]</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1000</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1001</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1010</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1011</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1100</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1101</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>1110</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
</tr>
<tr>
<td>1111</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
</tbody>
</table>

A = Analog input  
D = Digital I/O

**Note 1:** The POR value of the PCFG bits depends on the value of the PBADEN Configuration bit. When PBADEN = 1, PCFG<2:0> = 000; when PBADEN = 0, PCFG<2:0> = 111.

**Note 2:** AN5 through AN7 are available only on 40/44-pin devices.
A.2. Placa Open18f4520

Contenido obtenido de la web de WaveshareElectronics:

( http://www.wvshare.com/product/Open18F4520-Package-B.htm )

1. **PIC18F4520-I/P:** the high performance PIC MCU which features:
   - **Core:** PIC 8-bit RISC
   - **Operating Frequency:** 20MHz Max
   - **Operating Voltage:** 2-5.5V
   - **Package:** DIP40
   - **I/Os:** 33
   - **Memories:** 32kB Flash, 1536B RAM, 256B EEPROM
   - **Communication Interfaces:** 1 x MSSP(SPI/I2C), 1 x A/E/UART, 2 x PWM, 8 x ADC
   - **Debugging/Programming:** ICSP interfaces

2. **AMS1117-3.3:** 3.3V voltage regulator

3. **Power switch**
4. **Power indicator**
5. **LEDs**: convenient for indicating I/O status and/or program running state
6. **User keys**: for I/O input test and/or program control
7. **Joystick**: five positions
8. **Reset button**
9. **Buzzer**
10. **Infrared receiver**
11. **4M crystal oscillator**
12. **16 I/Os interface | 8-bit AD interface**
   o for connecting accessory boards which using I/O control, such as FT245 USB FIFO, 8 SEG LED, etc.
   o there's also 8-bit AD interface can be used for AD testing
13. **8 I/Os interface**: for connecting accessory boards which using I/O control, such as 8 Push Buttons, Motor, etc.
14. **SPI interface**: for connecting SPI peripherals, such as DataFlash (AT45DBxx), SD card, MP3, etc.
15. **I2C interface**: for connecting I2C peripherals, such as I/O expander (PCF8574), EEPROM (AT24Cxx), etc.
16. **1-WIRE interface**: for connecting 1-WIRE devices (TO-92 package), such as temperature sensor (DS18B20), electronic registration number (DS2401), etc.
17. **UART interface**: for connecting UART peripherals, such as RS232, RS485, USB TO UART, etc.
18. **PS/2 interface**: for connecting PS/2 keyboard and/or mouse
19. **Graphic multi-color LCD interface**: for connecting 2.2 inch multi-color touch screen LCD which using SPI control
20. **Character LCD interface**: for connecting character LCD, such as LCD1602 (3.3V blue backlight)
21. **5V DC jack**
22. **Custom crystal socket**
23. **VCC power input/output**: usually used for power supply output, and/or common ground with other application board
24. **MCU pins connector**: all the MCU pins are accessible on expansion connectors for further expansion
25. **ICSP interface**: for debugging/programming
26. **LEDs jumper**
27. **User keys jumper**
28. **Joystick jumper**
29. **PS/2 jumper**
30. **1-WIRE jumper**
31. **Buzzer jumper**
32. **IR jumper**
33. **VCC selection jumper**
34. **Crystal selection jumper**
35. **RA4 pull-up jumper**
ANEXO B. Comunicación SPI entre dos PIC18

B.1. Funciones para la comunicación SPI

#define nopSS_master LATAbits.LATA5
#define nopSS_slave PORTAbits.RA5
#define CS_master LATAbits.LATA5
#define CS_slave PORTAbits.RA5

void spi_mode(unsigned char CKP, unsigned char CKE, unsigned char SMP) // Sets SPI mode
{
    SSPCON1bits.CKP=CKP; // Clock Polarity
    SSPSTATbits.CKE=CKE; // Clock Select Bit, Clock Phase
    SSPSTATbits.SMP=SMP; // Sample Bit
}

void spi_interrupt_config (unsigned char enable, unsigned char priority)
{
    PIE1bits.SSPIE=enable;
    // 1: Enable the MSSP interrupt, 0: Disable the MSSP interrupt
    IPR1bits.SSPIP=priority;
    // 1: High Priority, 0: Low priority
}

void spi_enable_master(unsigned char clock) // Enable SSP port and set TRIS register. Sets SPI port as Master and sets Clock Frequency: 00: Fosc/4, 01: Fosc/16, 10: Fosc/64, 11: TMR2 output/2
{
    SSPCON1 = (SSPCON1 & 0xF0) | clock;
    // We change just the last 4 bits: ----00YY where clock=YY

    TRISAbits.TRISA5=0; // CS out
    TRISCbits.TRISC4=1; // SDI in
    TRISCbits.TRISC5=0; // SDO out

    SSPCON1bits.SSPEN=1; // Enable SPI port
}

void spi_enable_slave(unsigned char disable_nopSS) // Enable SSP port and set TRIS register
{
    SSPCON1 = (((SSPCON1 & 0xF0) + 0b0100) | disable_nopSS);
    // Disable_nopSS = 1: nopSS pin control disabled
    // Disable_nopSS = 0: nopSS pin control enabled

    TRISAbits.TRISA5=1; // CS in
    TRISCbits.TRISC4=1; // SDI in
    TRISCbits.TRISC5=0; // SDO out

    SSPCON1bits.SSPEN=1; // Enable SPI port
}
void spi_disable(void) // Enable SSP port and set TRIS register
{
    SSPCON1bits.SSPEN=0; // Disable SPI port
}

unsigned char spi_transfer(unsigned char x) // Write and Read a Byte on the SPI line
{
    SSPBUF = x;
    // We write a byte on the SSP buffer (MASTER MODE) --> The byte is transferred to the SSPSR --> The SSPBUF is deleted
    // We write a dummy data byte on the SSP buffer (SLAVE MODE) --> The byte is transferred to the SSPSR --> The SSPBUF is deleted
    while(SSPSTATbits.BF==0);
    // We wait until we receive a dummy data byte (MASTER MODE)
    // We wait until we receive a data byte (SLAVE MODE)
    return(SSPBUF);
    // We return the dummy byte we received (MASTER MODE)
    // We return the byte we received (SLAVE MODE)
}
B.2. Códigos para la comunicación entre dos PIC18

B.2.1. Código para el transmisor

```c
#include<p18f4520.h>
#include<delays.h>
#include"config.h"

// TRANSMISOR: envío de un byte al receptor que puede ser variado mediante los pulsadores S0 y S1, y que es mostrado en los LEDs

void main (void)
{
    unsigned char valor1, valor2, dummydata, i;
    valor1=0b11000000;
    valor2=0b00000011;
    i=6;

    ADCON0bits.ADON=0;
    ADCON1=0x0F;
    TRISB=0x00;
    TRISDbits.TRISD1=1;
    TRISDbits.TRISD0=1;

    nopSS_master=1;
    spi_mode(0,0,0);
    spi_interrupt_config (0,0);
    spi_enable_master(0b00);

    LATB=0b00011000;    // Secuencia de inicialización
    Delay10KTCYx(50);
    LATB=0b00100100;
    Delay10KTCYx(50);
    LATB=0b01000010;
    Delay10KTCYx(50);
    LATB=0b10000000;
    Delay10KTCYx(50);

    while(1)
    {
        PORTB=i;

        if(PORTDbits.RD1==0)
        {
            i++;
            Delay10TCYx(100);
            while(PORTDbits.RD1==0);
            Delay10TCYx(100);
        }
        else if(PORTDbits.RD0==0)
        {
            i--;
        }
    }
```
Delay10TCYx(100);
while(PORTDbits.RD0==0);
Delay10TCYx(100);
}
nopSS_master=0;
dummydata=spi_transfer(i);   // Envío de un valor útil
nopSS_master=1;
Delay10TCYx(10);            // Tiempo de refresco
}
spi_disable();
B.2.2. Código para el receptor

```c
#include<p18f4520.h>
#include<delays.h>
#include"config.h"

// RECEPTOR: recepción de un byte que se muestra en los LEDs
void main (void)
{
    unsigned char dummydata, valor;
    dummydata=0b11111111;
    valor=0b01010101;

    ADCON0bits.ADON=0;
    ADCON1=0x0F;
    TRISB=0x00;

    spi_mode(0,0,0);
    spi_interrupt_config (0,0,0);
    spi_enable_slave(0);

    LATB=0b00011000; // Secuencia de inicialización
    Delay10KTCYx(50);
    LATB=0b00100100;
    Delay10KTCYx(50);
    LATB=0b01000010;
    Delay10KTCYx(50);
    LATB=0b10000001;
    Delay10KTCYx(50);

    while(1)
    {
        valor=spi_transfer(dummydata); // Entrada de un valor útil
        LATB=valor; // Uso del valor útil en el seteo de los LEDS
        Delay10TCYx(10); // Tiempo de refresco
    }

    spi_disable();
}
```
ANEXO C. Códigos antecedentes para el nRF24L01

C.1. Código de WaveshareElectronics

/*
*==========================================================================================================
* File                :14L01.h
* Hardware Environment: OpenPIC16F877A && 1602 && 5v voltage && 4M crystal oscillator &&
* Build Environment   : MPLAB IDE
* Version             : V8.76
* By                  : Zhou Jie
* (c) Copyright 2011-2016, WaveShare
http://www.waveShare.net
All Rights Reserved
*==========================================================================================================
*/
#ifndef _24L01_H
#define _24L01_H

// CE Pin & CSN Pin & IRQ Pin
#define CE(x)      RA2=x
#define CSN(x)      RA3=x
void spi_init();
#define IRQ     RA4
#define u8    unsigned char

// SPI(nRF24L01) commands
#define READ_REG_NRF24L01     0x00     // Define read command to register
#define WRITE_REG_NRF24L01    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

//SPI(nRF24L01) registers(addresses)
#define CONFIGA         0x00    // 'Config' register address
#define EN_AA          0x01                  // 'Enable Auto Acknowledgment' register address

#endif
*/
#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

void RX_Mode(void);
void TX_Mode(u8 * tx_buf);
void nRF24L01_Initial(void);
void nRF24L01_Config(void);
void nRF24L01_Send(void);
void nRF24L01_Receive(void);

#endif /*_NRF24L01_H*/
rief
*==========================================================================================
* File                : 24l01.c
* Hardware Environment:
&&
* Build Environment   : MPLAB IDE
* Version             : V8.76
* By                  : Zhou Jie
* *
(c) Copyright 2011-2016, WaveShare
http://www.waveShare.net
All Rights Reserved
*==========================================================================================*/

#include <pic.h>
#include "24L01.h"

#define TX_ADR_WIDTH 5
#define TX_PLOAD_WIDTH 32
#define u8 unsigned char
#define RX_DR   0x40
#define TX_DS   0x20
#define MAX_RT   0x10
#define u16 unsigned int

u8 TX_ADDRESS[TX_ADR_WIDTH] = {0xb2,0xb2,0xb3,0xb4,0x01};
u8 RX_BUF[TX_PLOAD_WIDTH];
u8 TX_BUF[TX_PLOAD_WIDTH]={
  0x01,0x02,0x03,0x4,0x05,0x06,0x07,0x08,
  0x09,0x10,0x11,0x12,0x13,0x14,0x15,0x16,
  0x17,0x18,0x19,0x20,0x21,0x22,0x23,0x24,
  0x25,0x26,0x27,0x28,0x29,0x30,0x31,0x32};

void spi_init()
{
  ADCON1=0X07;
  TRISA=0X10;
  TRISB=0X0;
  TRISC=0X10;

  SSPSTAT=0xC0;
  SSPCON=0x20;
  INTCON=0x00;
  PIR1=0x00;
}

void SPI_Send_byte(u8 data)
{
  SSPBUF=data;
  while(!SSPIF);
  SSPIF=0;
}

static u8 SPI_Receive_byte(u8 data)
{
  u8 temp;
  SSPBUF=data;
}
while(!SSPIF);
    SSPIF=0;
    temp=SSPBUF;
    return temp;
}

static void delay1us(u16 t)
{
    while(--t);
}

u8 SPI_RW_Reg(u8 reg,u8 value)
{
    u8 status;
    CSN(0);
    status=SPI_Receive_byte(reg); //select register and write value to it
    SPI_Send_byte(value);
    CSN(1);
    return(status);
}

u8 SPI_Read_Reg(u8 reg)
{
    u8 status;
    CSN(0);
    SPI_Send_byte(reg);
    status=SPI_Receive_byte(0); //select register and write value to it
    CSN(1);
    return(status);
}

u8 SPI_Read_Buf(u8 reg,u8 *pBuf,u8 bytes)
{
    u8 status,byte_ctr;
    CSN(0);
    status=SPI_Receive_byte(reg);
    for(byte_ctr=0;byte_ctr<bytes;byte_ctr++)
    {
        pBuf[byte_ctr]=SPI_Receive_byte(0);
    }
    CSN(1);
    return(status);
}

u8 SPI_Write_Buf(u8 reg,u8 *pBuf,u8 bytes)
{
    u8 status,byte_ctr;
    CSN(0);
    status=SPI_Receive_byte(reg);
    delay1us(1);
    for(byte_ctr=0;byte_ctr<bytes;byte_ctr++)
    {
        SPI_Send_byte(*pBuf++);
    }
    CSN(1);
    return(status);
}

u8 nRF24L01_RxPacket(u8* rx_buf)
{
    u8 status,revale=0;
    CE(0);
    delay1us(10);
status=SPI_Receive_byte(STATUS);
if(status & RX_DR)
{
    // CE(1);
    SPI_Read_Buf(RD_RX_PLOAD, rx_buf, TX_PLOAD_WIDTH); // read receive payload from RX_FIFO buffer
    // CE(0);
    revale =1;
}
SPI_RW_Reg(WRITE_REG_NRF24L01 + STATUS, status);  
CE(1);
return revale;
}

void nRF24L01_TxPacket(unsigned char * tx_buf)
{
    CE(0);
    SPI_Write_Buf(WRITE_REG_NRF24L01 + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH);
    SPI_Write_Buf(WR_TX_PLOAD, tx_buf, TX_PLOAD_WIDTH);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RX_PW_P0, TX_PLOAD_WIDTH);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RX_PW_P0, TX_PLOAD_WIDTH);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + CONFIGA, 0x0e);
    CE(1);
    delay1us(10);
}

void RX_Mode(void)
{
    CE(0);
    SPI_Write_Buf(WRITE_REG_NRF24L01 + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RX_PW_P0, TX_PLOAD_WIDTH);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + EN_AA, 0x3f);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + EN_RXADDR, 0x3f);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RF_CH, 40);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RF_SETUP, 0x07);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + CONFIGA, 0x0f);
    CE(1);
}

void TX_Mode(u8 * tx_buf)
{
    CE(0);
    SPI_Write_Buf(WRITE_REG_NRF24L01 + TX_ADDR, TX_ADDRESS, TX_ADR_WIDTH);
    SPI_Write_Buf(WRITE_REG_NRF24L01 + RX_ADDR_P0, TX_ADDRESS, TX_ADR_WIDTH);
    SPI_Write_Buf(WR_TX_PLOAD, tx_buf, TX_PLOAD_WIDTH);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + EN_AA, 0x3f);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + EN_RXADDR, 0x3f);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + SETUP_RETR, 0x0a);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RF_CH, 40);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RF_SETUP, 0x07);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RX_PW_P0, TX_PLOAD_WIDTH);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RX_PW_P0, TX_PLOAD_WIDTH);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + EN_AA, 0x3f);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RF_PW_P0, 0x07);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RX_PW_P0, TX_PLOAD_WIDTH);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RX_PW_P0, TX_PLOAD_WIDTH);
    CE(1);
    delay1us(10);
}

void nRF24L01_Initial(void)
{
}
void nRF24L01_Config(void)
{
    // initial io
    // CE(0);     // CE=0 ; chip enable
    // CSN(1);    // CSN=1  Spi disable
    SPI_RW_Reg(WRITE_REG_NRF24L01 + CONFIGA, 0x0e); // Set PWR_UP bit, enable CRC(2 bytes) & Prim:RX. RX_DR enabled..
    SPI_RW_Reg(WRITE_REG_NRF24L01 + EN_AA, 0x3f);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + EN_RXADDR, 0x3f); // Enable Pipe0
    // SPI_RW_Reg(WRITE_REG_NRF24L01 + SETUP_AW, 0x02); // Setup address width=5 bytes
    // SPI_RW_Reg(WRITE_REG_NRF24L01 + SETUP_RETR, 0x1a); // 500us + 86us, 10 retrans...
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RF_CH, 40);
    SPI_RW_Reg(WRITE_REG_NRF24L01 + RF_SETUP, 0x07); // TX_PWR:0dBm, Datarate:2Mbps,
}

void NRF24L01_Send(void)
{
    u8 status=0x00;
    // CSN(0);
    TX_Mode(TX_BUF);
    while(IRQ);
    CE(0);
    delay1us(10);
    status=SPI_Read_Reg(STATUS);
    if(status&TX_DS) /*tx_ds == 0x20*/
    {
        PORTB=0x0F;
        SPI_RW_Reg(WRITE_REG_NRF24L01 + STATUS, 0x20);
    }
    else if(status&MAX_RT)
    {
        PORTB=0xFF;
        SPI_RW_Reg(WRITE_REG_NRF24L01 + STATUS, 0x10);
    }
    // status=20;
}

void NRF24L01_Receive(void)
{
    u8 i, status=0x01;
    RX_Mode();
    while(IRQ);
    CE(0);
    delay1us(10);
    status=SPI_Read_Reg(STATUS);
    if(status & 0x40)
    {PORTB=0x33;
        CE(1);
        SPI_Read_Buf(RD_RX_PLOAD, RX_BUF, TX_PLOAD_WIDTH);// read receive payload from RX_FIFO buffer
    }
}
for(i=0;i<32;i++)
{
    PORTB=RX_BUF[i];TXREG=RX_BUF[i];delay1us(3000);
}
SPI_RW_Reg(WRITE_REG_NRF24L01 + STATUS, 0x40);
CE(1);

#include <pic.h>
#include "24L01.h"
#include <pic16f876a.h>

#define T_O_R 1
#define T_O_R 0
#define u16 unsigned int

void uart_int()
{
    SPBRG=0x8C;
    TXSTA=0x24;
    RCSTA=0x90;
    TXIE=0x0;
    GIE=0x1;
    PEIE=0x1;
}
void delay_ms(u16 ms)
{
    while(--ms);
}
void main(void)
{
    spi_init();
    uart_int();
    nRF24L01_Initial();
    PORTB=0;
while(1)
{
    if(T_O_R)
    {
        //TX_Mode();
        delay_ms(300);
        NRF24L01_Send();
    }
    else
    {
        NRF24L01_Receive();
    }
}
C.2. Código para AVR de Google Code

```c
#include <stdio.h>
#include <string.h>
#include <avr/io.h>
#include "nrf24l01.h"

/*NRF commands*/
#define R_REGISTER          (0x00)      /*read register*/
#define W_REGISTER          (0x20)      /*write register*/
#define R_RX_PAYLOAD        (0x61)      /*read RX payload*/
#define W_TX_PAYLOAD        (0xA0)      /*write TX payload*/
#define FLUSH_TX            (0xE1)      /*flush TX FIFO*/
#define FLUSH_RX            (0xE2)      /*flush RX FIFO*/

/*NRF registers*/
#define CONFIG              (0x00)      /*configuration*/
#define EN_AA               (0x01)      /*enable automatic ACL*/
#define EN_RXADDR           (0x02)      /*enable RX addresses*/
#define SETUP_AW            (0x03)      /*setup address width*/
#define RETR                (0x04)      /*setup retransmission*/
#define RF_CH               (0x05)      /*RF channel*/
#define RF_SETUP            (0x06)      /*RF setup*/
#define STATUS              (0x07)      /*setup retransmission*/
#define OBSERVE_TX          (0x08)      /*TX observe*/
#define RPD                 (0x09)      /*received power detector*/
#define RX_ADDR_P0          (0x0A)      /*receive address data pipe 0*/
#define RX_ADDR_P1          (0x0B)      /*receive address data pipe 1*/
#define RX_ADDR_P2          (0x0C)      /*receive address data pipe 2*/
#define RX_ADDR_P3          (0x0D)      /*receive address data pipe 3*/
#define RX_ADDR_P4          (0x0E)      /*receive address data pipe 4*/
#define RX_ADDR_P5          (0x0F)      /*receive address data pipe 5*/
#define TX_ADDR             (0x10)      /*transmit address*/
#define RX_PW_P0            (0x11)      /*pipe 0 payload length*/
#define RX_PW_P1            (0x12)      /*pipe 1 payload length*/
#define RX_PW_P2            (0x13)      /*pipe 2 payload length*/
#define RX_PW_P3            (0x14)      /*pipe 3 payload length*/
#define RX_PW_P4            (0x15)      /*pipe 4 payload length*/
#define RX_PW_P5            (0x16)      /*pipe 5 payload length*/
#define FIFO_STATUS         (0x17)      /*FIFO status*/
#define DYNPD               (0x1C)      /*enable dynamic payload length*/

/*CONFIG register bits*/
#define PRIM_RX             (1<<0)
#define PWR_UP              (1<<1)
#define CRCO                (1<<2)
#define EN_CRC              (1<<3)
#define MASK_MAX_RT         (1<<4)
#define MASK_TX_DS          (1<<5)
#define MASK_RX_DR          (1<<6)

/*STATUS register bits*/
#define ST_TX_FULL          (1<<0)
#define MAX_RT              (1<<4)
#define TX_DS               (1<<5)
#define RX_DR               (1<<6)
```
/*FIFO_STATUS register bits*/
#define RX_EMPTY      (1<<0)
#define RX_FULL       (1<<1)
#define TX_EMPTY      (1<<4)
#define TX_FULL       (1<<5)
#define TX_REUSE      (1<<6)
/*timing definitions*/
#define POWONRST      (100/*ms*/) /*power on reset*/
#define TPD2STBY      (5/*ms*/)   /*powerdown to standby*/
/*avr SPI*/
#define DDR_SPI       DDRB
#define DD_CE         DDB0
#define DD_SSEL       DDB2
#define DD_MOSI       DDB3
#define DD_MISO       DDB4
#define DD_SCLK       DDB5
/*GPIO macros*/
#define CSN_LO()      PORTB &= ~(1<<DD_SSEL)
#define CSN_HI()      PORTB |=  (1<<DD_SSEL)
#define CE_LO()       PORTB &= ~(1<<DD_CE)
#define CE_HI()       PORTB |=  (1<<DD_CE)

uint8_t spi_read_byte()
{
    SPDR = 0xFF;
    while (!(SPSR & (1<<SPIF)));
    return SPDR;
}

uint8_t spi_write_byte(uint8_t w)
{
    SPDR = w;
    while (!(SPSR & (1<<SPIF)));
    return SPDR;
}

uint8_t nrf_read_reg(uint8_t reg)
{
    CSN_LO();
    spi_write_byte(R_REGISTER|reg);
    reg = spi_read_byte();
    CSN_HI();
    return reg;
}

void nrf_write_reg(uint8_t reg, uint8_t val)
{
    CSN_LO();
    spi_write_byte(W_REGISTER|reg);
    spi_write_byte(val);
    CSN_HI();
}

void nrf_write_multibyte_reg(uint8_t reg, uint8_t *buf, uint8_t len)
{
    CSN_LO();
    spi_write_byte(W_REGISTER|reg);
    while (len)
    {
        spi_write_byte(buf[len--]);
    }
    CSN_HI();
}
void nrf_read_multibyte_reg(uint8_t reg, uint8_t *buf, uint8_t len)
{
    CSN_LO();
    spi_write_byte(R_REGISTER|reg);
    while (len--)
    {
        *buf++ = spi_read_byte();
    }
    CSN_HI();
} 

void nrf_init()
{
    /*set MOSI, SCLK, SSEL, CE output*/
    DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCLK)|(1<<DD_SSEL)|(1<<DD_CE);
    CE_LO();
    CSN_HI();
    /*enable SPI, Master*/
    SPCR = (1<<SPE)|(1<<MSTR);
    delay_ms(POWONRST);
    /*init nordic chip*/
    nrf_write_reg(CONFIG, 0);
    nrf_write_reg(RF_SETUP, 0x08); //TODO
    nrf_enable_pipe(NRF_PIPE_ALL, false);
    nrf_enable_autoack(NRF_PIPE_ALL, false);
    nrf_write_reg(CONFIG, (MASK_TX_DS|MASK_RX_DR|MASK_MAX_RT|PWR_UP));
    delay_ms(TPD2STBY);
    /*should now enter standby mode*/
}

void nrf_clear_int()
{
    uint8_t st;
    st = nrf_read_reg(STATUS);
    nrf_write_reg(STATUS, (st | 0x70));
}

void nrf_set_operation_mode(nrf_operation_mode_t op_mode)
{
    uint8_t config;
    config = nrf_read_reg(CONFIG);
    switch (op_mode) {
        case NRF_PRIM_TX:
            nrf_write_reg(CONFIG, (config & ~PRIM_RX));
            break;
        case NRF_PRIM_RX:
            nrf_write_reg(CONFIG, (config | PRIM_RX));
            break;
    }
}

void nrf_set_power_mode(nrf_power_mode_t pwr_mode)
{
    uint8_t config;
    config = nrf_read_reg(CONFIG);
    switch (pwr_mode) {
        case NRF_POWER_DOWN:
            nrf_write_reg(CONFIG, (config & ~PWR_UP));
            break;
        case NRF_POWER_UP:
            nrf_write_reg(CONFIG, (config | PWR_UP));
            break;
    }
}
void nrf_enable_pipe(nrf_pipe_t pipe, bool enable)
{
    uint8_t pipes;
    pipes = nrf_read_reg(EN_RXADDR);
    if (enable) {
        pipes |= pipe;
    } else {
        pipes &= ~pipe;
    }
    nrf_write_reg(EN_RXADDR, pipes);
}

void nrf_enable_autoack(nrf_pipe_t pipe, bool enable)
{
    uint8_t pipes;
    pipes = nrf_read_reg(EN_AA);
    if (enable) {
        pipes |= pipe;
    } else {
        pipes &= ~pipe;
    }
    nrf_write_reg(EN_AA, pipes);
}

void nrf_set_pipe_width(nrf_pipe_t pipe, int pw)
{
    uint8_t reg=0;
    switch (pipe) {
        case NRF_PIPE_0:
            reg = RX_PW_P0);
            break;
        case NRF_PIPE_1:
            reg = RX_PW_P1;
            break;
        case NRF_PIPE_2:
            reg = RX_PW_P2;
            break;
        case NRF_PIPE_3:
            reg = RX_PW_P3;
            break;
        case NRF_PIPE_4:
            reg = RX_PW_P4;
            break;
        case NRF_PIPE_5:
            reg = RX_PW_P5;
            break;
        case NRF_PIPE_ALL;/*TODO*/
            break;
    }
    nrf_write_reg(reg, pw);
}

void nrf_set_tx_address(uint8_t *addr)
{
    nrf_write_multibyte_reg(TX_ADDR, addr, 5);
}

void nrf_flush_tx()
{
}
Comunicación por RF entre microcontroladores PIC18 mediante el módulo NRF24L01

#include <avr/io.h>
#include "nrf24l01.h"

#define LED_PIN (1<<0)

void toggle_led()
{
    PORTD |= LED_PIN;
    delay_ms(500);
    PORTD &= ~(LED_PIN);
    delay_ms(500);
}

int main (void)
{
    char buf[32];
    memset(buf, 'h', 32);

    //set LED_PIN to output
    DDRD |= LED_PIN;

    //init nordic chip
    nrf_init();
    nrf_enable_pipe(NRF_PIPE_0, true);
    nrf_set_pipe_width(NRF_PIPE_0, 32);

    while (1) {
        nrf_send(buf, 32);
        toggle_led();
    }
    return 0;
}
ANEXO D. Información del nRF24L01

D.1. Esquemático y layout del módulo NRF24L01 RF Board
D.2. **Datasheet del nRF24L01**

**PRELIMINARY PRODUCT SPECIFICATION**

**nRF24L01**

**Single chip 2.4 GHz Transceiver**

**FEATURES**
- True single chip GFSK transceiver
- Complete OSI Link Layer in hardware
- Enhanced ShockBurst™
- Auto ACK & retransmit
- Address and CRC computation
- On the air data rate 1 or 2Mbps
- Digital interface (SPI) speed 0-10 Mbps
- 125 RF channel operation
- Short switching time enable frequency hopping
- Fully RF compatible with nRF24XX
- 5V tolerant signal input pads
- 20-pin package (QFN20 4x4mm)
- Uses ultra low cost +/- 60 ppm crystal
- Uses low cost chip inductors and 2-layer PCB
- Power supply range: 1.9 to 3.6 V

**APPLICATIONS**
- Wireless mouse, keyboard, joystick
- Keyless entry
- Wireless data communication
- Alarm and security systems
- Home automation
- Surveillance
- Automotive
- Telemetry
- Intelligent sports equipment
- Industrial sensors
- Toys

**GENERAL DESCRIPTION**

nRF24L01 is a single chip radio transceiver for the world wide 2.4 - 2.5 GHz ISM band. The transceiver consists of a fully integrated frequency synthesizer, a power amplifier, a crystal oscillator, a demodulator, modulator and Enhanced ShockBurst™ protocol engine. Output power, frequency channels, and protocol setup are easily programmable through a SPI interface. Current consumption is very low, only 9.0mA at an output power of +6dBm and 12.3mA in RX mode. Built-in Power Down and Standby modes makes power saving easily realizable.

**QUICK REFERENCE DATA**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum supply voltage</td>
<td>1.9</td>
<td>V</td>
</tr>
<tr>
<td>Maximum output power</td>
<td>0</td>
<td>dBm</td>
</tr>
<tr>
<td>Maximum data rate</td>
<td>2000</td>
<td>kbps</td>
</tr>
<tr>
<td>Supply current in TX mode @ 0dBm output power</td>
<td>11.3</td>
<td>mA</td>
</tr>
<tr>
<td>Supply current in RX mode @ 2000 kbps</td>
<td>12.3</td>
<td>mA</td>
</tr>
<tr>
<td>Temperature range</td>
<td>-40 to +85</td>
<td>°C</td>
</tr>
<tr>
<td>Sensitivity @ 1000 kbps</td>
<td>-85</td>
<td>dBm</td>
</tr>
<tr>
<td>Supply current in Power Down mode</td>
<td>900</td>
<td>nA</td>
</tr>
</tbody>
</table>

Table 1 nRF24L01 quick reference data
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

<table>
<thead>
<tr>
<th>Type Number</th>
<th>Description</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>nRF24L01</td>
<td>20 pin QFN 4x4, lead free (green)</td>
<td>A</td>
</tr>
<tr>
<td>nRF24L01-16C</td>
<td>Bare Die</td>
<td>A</td>
</tr>
<tr>
<td>nRF24L01-EVKIT</td>
<td>Evaluation kit (2 test PCB, 2 configuration PCB, SW)</td>
<td>1.0</td>
</tr>
</tbody>
</table>

Table 2 nRF24L01 ordering information

BLOCK DIAGRAM

Figure 1 nRF24L01 with external components.
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

PIN FUNCTIONS

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Pin function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CE</td>
<td>Digital Input</td>
<td>Chip Enable Activates RX or TX mode</td>
</tr>
<tr>
<td>2</td>
<td>CSN</td>
<td>Digital Input</td>
<td>SPI Chip Select</td>
</tr>
<tr>
<td>3</td>
<td>SCK</td>
<td>Digital Input</td>
<td>SPI Clock</td>
</tr>
<tr>
<td>4</td>
<td>MOSI</td>
<td>Digital Input</td>
<td>SPI Slave Data Input</td>
</tr>
<tr>
<td>5</td>
<td>MISO</td>
<td>Digital Output</td>
<td>SPI Slave Data Output, with tri-state option</td>
</tr>
<tr>
<td>6</td>
<td>IRQ</td>
<td>Digital Output</td>
<td>Maskable interrupt pin</td>
</tr>
<tr>
<td>7</td>
<td>VDD</td>
<td>Power</td>
<td>Power Supply (+3V DC)</td>
</tr>
<tr>
<td>8</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>9</td>
<td>XC2</td>
<td>Analog Output</td>
<td>Crystal Pin 2</td>
</tr>
<tr>
<td>10</td>
<td>XC1</td>
<td>Analog Input</td>
<td>Crystal Pin 1</td>
</tr>
<tr>
<td>11</td>
<td>VDD_PA</td>
<td>Power Output</td>
<td>Power Supply (+1.8V) to Power Amplifier</td>
</tr>
<tr>
<td>12</td>
<td>ANT1</td>
<td>RF</td>
<td>Antenna interface 1</td>
</tr>
<tr>
<td>13</td>
<td>ANT2</td>
<td>RF</td>
<td>Antenna interface 2</td>
</tr>
<tr>
<td>14</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>15</td>
<td>VDD</td>
<td>Power</td>
<td>Power Supply (+3V DC)</td>
</tr>
<tr>
<td>16</td>
<td>IREF</td>
<td>Analog Input</td>
<td>Reference current</td>
</tr>
<tr>
<td>17</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
<tr>
<td>18</td>
<td>VDD</td>
<td>Power</td>
<td>Power Supply (+3V DC)</td>
</tr>
<tr>
<td>19</td>
<td>DVDD</td>
<td>Power Output</td>
<td>Positive Digital Supply output for de-coupling purposes</td>
</tr>
<tr>
<td>20</td>
<td>VSS</td>
<td>Power</td>
<td>Ground (0V)</td>
</tr>
</tbody>
</table>

Table 3. nRF24L01 pin function

PIN ASSIGNMENT

Figure 2. nRF24L01 pin assignment (top view) for a QFN20 4x4 package.
## PRELIMINARY PRODUCT SPECIFICATION

**nRF24L01 Single Chip 2.4 GHz Radio Transceiver**

### ELECTRICAL SPECIFICATIONS

Conditions: VDD = +3V, VSS = 0V, T_A = -40°C to +85°C

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter (condition)</th>
<th>Notes</th>
<th>Min.</th>
<th>Typ.</th>
<th>Max.</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>Supply voltage</td>
<td></td>
<td>1.9</td>
<td>3.0</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>TEMP</td>
<td>Operating Temperature</td>
<td></td>
<td>-40</td>
<td>-27</td>
<td>-85</td>
<td>°C</td>
</tr>
<tr>
<td>V_H</td>
<td>HIGH level input voltage</td>
<td>1</td>
<td>0.7VDD</td>
<td>5.25</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>V_L</td>
<td>LOW level input voltage</td>
<td></td>
<td>VSS</td>
<td>0.3VDD</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>V_H0M</td>
<td>HIGH level output voltage (I_{out}=0.5mA)</td>
<td>VDD-0.3</td>
<td>VDD</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V_L0M</td>
<td>LOW level output voltage (I_{out}=0.5mA)</td>
<td>VSS</td>
<td>0.3</td>
<td>V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f_{op}</td>
<td>Operating frequency</td>
<td>2</td>
<td>2400</td>
<td></td>
<td>2525</td>
<td>MHz</td>
</tr>
<tr>
<td>f_{XTAL}</td>
<td>Crystal frequency</td>
<td></td>
<td>16</td>
<td></td>
<td></td>
<td>MHz</td>
</tr>
<tr>
<td>f_{DM}</td>
<td>Frequency deviation @ 1000kbps</td>
<td></td>
<td>±160</td>
<td></td>
<td></td>
<td>kHz</td>
</tr>
<tr>
<td>f_{PM}</td>
<td>Frequency deviation @ 2000kbps</td>
<td></td>
<td>±320</td>
<td></td>
<td></td>
<td>kHz</td>
</tr>
<tr>
<td>R_{Ss}</td>
<td>Data rate ShockBurstTM</td>
<td></td>
<td>0</td>
<td></td>
<td>2000</td>
<td>kbps</td>
</tr>
<tr>
<td>f_{CHAN1}</td>
<td>Channel spacing @ 1000kbps</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>MHz</td>
</tr>
<tr>
<td>f_{CHAN2}</td>
<td>Channel spacing @ 2000kbps</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>MHz</td>
</tr>
<tr>
<td>P_{max}</td>
<td>Maximum Output Power</td>
<td>3</td>
<td>0</td>
<td></td>
<td>+4</td>
<td>dBm</td>
</tr>
<tr>
<td>P_{PCC}</td>
<td>RF Power Control Range</td>
<td></td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>dB</td>
</tr>
<tr>
<td>P_{REC}</td>
<td>RF Power Accuracy</td>
<td></td>
<td>±4</td>
<td></td>
<td></td>
<td>dB</td>
</tr>
<tr>
<td>P_{BW}</td>
<td>20dB Bandwidth for Modulated Carrier (2000kbps)</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td>kHz</td>
</tr>
<tr>
<td>P_{P1}</td>
<td>1^st Adjacent Channel Transmit Power 2MHz</td>
<td>-20</td>
<td></td>
<td></td>
<td></td>
<td>dBm</td>
</tr>
<tr>
<td>P_{P2}</td>
<td>2^nd Adjacent Channel Transmit Power 4MHz</td>
<td>-50</td>
<td></td>
<td></td>
<td></td>
<td>dBm</td>
</tr>
<tr>
<td>I_{SDD}</td>
<td>Supply current @ 0dBm output power</td>
<td></td>
<td>11.3</td>
<td></td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>I_{SDD}</td>
<td>Supply current @ -15dBm output power</td>
<td>7.6</td>
<td></td>
<td></td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>I_{VDD}</td>
<td>Average Supply current @ -4dBm output power, Enhanced ShockBurstTM</td>
<td>0.05</td>
<td></td>
<td></td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>I_{SDD}</td>
<td>Supply current in Standby-1 mode</td>
<td>32</td>
<td></td>
<td></td>
<td></td>
<td>µA</td>
</tr>
<tr>
<td>I_{SDD}</td>
<td>Supply current in power down</td>
<td>900</td>
<td></td>
<td></td>
<td></td>
<td>nA</td>
</tr>
</tbody>
</table>

1. All digital inputs handle up to 5.25V signal inputs. Keep in mind that the VDD of the nRF24L01 must match the V_{DD} of the driving device for output pins.
2. Usable bandwidth determined by local regulations.
3. Antenna load impedance = 100Ω+ j175Ω
4. Antenna load impedance = 100Ω+ j175Ω. Effective data rate 1000kbps or 2000 kbps
5. Antenna load impedance = 100Ω+ j175Ω. Effective data rate 10kbps and full packets

---

Nordic Semiconductor ASA - Vestre Rosten 81, N-7075 Tiller, Norway - Phone +4772898900 - Fax +4772898989
Revision: 1.1 Page 4 of 38 November 2005
## PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

| Receiver operation |  |  |  |
|--------------------|-----------------------------|------------------|
| I_{DDQ} ( Supply current one channel 2000kbps ) | 12.3 | mA |
| I_{DDQ} ( Supply current one channel 1000kbps ) | 11.8 | mA |
| RX_{SENS} ( Sensitivity at 0.1%BER (@2000kbps) ) | -82 | dBm |
| RX_{SENS} ( Sensitivity at 0.1%BER (@1000kbps) ) | -85 | dBm |
| C/I_{50} ( C/I Co-channel ) | 5^{6} | dB |
| C/I_{1ST} ( 1st Adjacent Channel Selectivity C/I 2MHz ) | 7^{5} | dB |
| C/I_{2ND} ( 2nd Adjacent Channel Selectivity C/I 4MHz ) | 7^{-19} | dB |
| C/I_{3RD} ( 3rd Adjacent Channel Selectivity C/I 6MHz ) | 7^{-27} | dB |
| C/I_{CO} ( C/I Co-channel ) | 7^{-14} | dB |
| C/I_{50} ( C/I Co-channel ) | 7^{-14} | dB |
| C/I_{1ST} ( 1st Adjacent Channel Selectivity C/I 1MHz ) | 7^{-10} | dB |
| C/I_{2ND} ( 2nd Adjacent Channel Selectivity C/I 2MHz ) | 7^{-22} | dB |
| C/I_{3RD} ( 3rd Adjacent Channel Selectivity C/I 3MHz ) | 7^{-30} | dB |

Table 4 nRF24L01 RF specifications

---

6 Data rate is 2000kbps for the following C/I measurements
7 According to ETSI EN 300 220-1 V1.3.1 (2000-09) page 36
8 nRF24L01 equal modulation on interfering signal
9 Data rate is 1000kbps for the following C/I measurements
10 According to ETSI EN 300 220-1 V1.3.1 (2000-09) page 36
11 nRF24L01 equal modulation on interfering signal

Nordic Semiconductor ASA - Vestre Rosten 81, N-7075 Tiller, Norway - Phone +4772898900 - Fax +4772898989
Revision: 1.1 Page 5 of 38 November 2005
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

PACKAGE OUTLINE

nRF24L01 uses the QFN20 4x4 package, with matt tin plating.

TOP VIEW

SIDE VIEW
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

Figure 3 nRF24L01 Package Outline.

<table>
<thead>
<tr>
<th>Package Type</th>
<th>A</th>
<th>A1</th>
<th>A3</th>
<th>K</th>
<th>D/E</th>
<th>e</th>
<th>D2/E2</th>
<th>L</th>
<th>L1</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAW QFN20</td>
<td>0.80</td>
<td>0.00</td>
<td>0.20</td>
<td>0.20</td>
<td>4.0 BSC(^\text{12})</td>
<td>0.5 BSC</td>
<td>2.50</td>
<td>0.35</td>
<td>0.18</td>
<td></td>
</tr>
<tr>
<td>(4x4 mm)</td>
<td>0.85</td>
<td>0.02</td>
<td>0.05</td>
<td>REF.</td>
<td>min</td>
<td>2.60</td>
<td>0.40</td>
<td>0.15</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.70</td>
<td>0.45</td>
<td>max</td>
<td>0.30</td>
<td></td>
</tr>
</tbody>
</table>

\(^{12}\) BSC: Basic Spacing between Centers, ref. JEDEC standard 55, page 4.17-11/A
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

Package marking:

<table>
<thead>
<tr>
<th>n</th>
<th>R</th>
<th>F</th>
<th>B</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>Y</td>
<td>Y</td>
<td>W</td>
<td>W</td>
<td>L</td>
</tr>
</tbody>
</table>

Abbreviations:
- DDDDD – Product number, e.g. 24L01
- B – Build Code, i.e. unique code for production sites, package type and test platform
- X – "X" grade, i.e. Engineering Samples (optional)
- YY – 2 digit Year number
- WW – 2 digit Week number
- LL – 2 letter wafer lot number code

Absolute Maximum Ratings

Supply voltages
- VDD ......................... - 0.3V to + 3.6V
- VSS .............................. 0V

Input voltage
- VI ............................. - 0.3V to 5.25V

Output voltage
- VO .............................. VSS to VDD

Total Power Dissipation
- PD(TA=85°C) ......................... 60mW

Temperatures
- Operating Temperature .... - 40°C to + 85°C
- Storage Temperature ....... - 40°C to + 125°C

Note: Stress exceeding one or more of the limiting values may cause permanent damage to the device.

ATTENTION!
Electrostatic Sensitive Device
Observe Precaution for handling.
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

Glossary of Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>ART</td>
<td>Auto Re-Transmit</td>
</tr>
<tr>
<td>CE</td>
<td>Chip Enable</td>
</tr>
<tr>
<td>CLK</td>
<td>Clock</td>
</tr>
<tr>
<td>CRC</td>
<td>Cyclic Redundancy Check</td>
</tr>
<tr>
<td>CSN</td>
<td>Chip Select NOI</td>
</tr>
<tr>
<td>ESB</td>
<td>Enhanced ShockBurst™</td>
</tr>
<tr>
<td>GFSK</td>
<td>Gaussian Frequency Shift Keying</td>
</tr>
<tr>
<td>IRQ</td>
<td>Interrupt Request</td>
</tr>
<tr>
<td>ISM</td>
<td>Industrial-Scientific-Medical</td>
</tr>
<tr>
<td>LNA</td>
<td>Low Noise Amplifier</td>
</tr>
<tr>
<td>LSB</td>
<td>Least Significant Bit</td>
</tr>
<tr>
<td>LSByte</td>
<td>Least Significant Byte</td>
</tr>
<tr>
<td>Mbps</td>
<td>Megabit per second</td>
</tr>
<tr>
<td>MCU</td>
<td>Micro Controller Unit</td>
</tr>
<tr>
<td>MISO</td>
<td>Master In Slave Out</td>
</tr>
<tr>
<td>MOSI</td>
<td>Master Out Slave In</td>
</tr>
<tr>
<td>MSB</td>
<td>Most Significant Bit</td>
</tr>
<tr>
<td>MSByte</td>
<td>Most Significant Byte</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PER</td>
<td>Packet Error Rate</td>
</tr>
<tr>
<td>PID</td>
<td>Packet Identity Bits</td>
</tr>
<tr>
<td>PLD</td>
<td>Payload</td>
</tr>
<tr>
<td>PRX</td>
<td>Primary RX</td>
</tr>
<tr>
<td>PTX</td>
<td>Primary TX</td>
</tr>
<tr>
<td>PWR_DWN</td>
<td>Power Down</td>
</tr>
<tr>
<td>PWR_UP</td>
<td>Power Up</td>
</tr>
<tr>
<td>RX</td>
<td>Receive</td>
</tr>
<tr>
<td>RX_DR</td>
<td>Receive Data Ready</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>TX</td>
<td>Transmit</td>
</tr>
<tr>
<td>TX_DS</td>
<td>Transmit Data Sent</td>
</tr>
</tbody>
</table>

Table 5 Glossary
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

FUNCTIONAL DESCRIPTION

Modes of operation
The nRF24L01 can be set in the following main modes depending on the level of the following primary I/Os and configuration registers:

<table>
<thead>
<tr>
<th>Mode</th>
<th>PWR_UP register</th>
<th>PRIM_RX register</th>
<th>CE</th>
<th>FIFO state</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX mode</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Data in TX FIFO</td>
</tr>
<tr>
<td>TX mode</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 stays in TX mode until packet transmission is finished</td>
</tr>
<tr>
<td>TX mode</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>TX FIFO empty</td>
</tr>
<tr>
<td>Standby-II</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>No ongoing packet transmission</td>
</tr>
<tr>
<td>Standby-I</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>Power Down</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 6: nRF24L01 main modes

An overview of the nRF24L01 I/O pins in different modes is given in Table 7.

Pin functions in the different modes of nRF24L01

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Direction</th>
<th>TX Mode</th>
<th>RX Mode</th>
<th>Standby Modes</th>
<th>Power Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>Input</td>
<td>High Pulse &gt;10µs</td>
<td>High</td>
<td>Low</td>
<td>-</td>
</tr>
<tr>
<td>CSN</td>
<td>Input</td>
<td>SPI Chip Select, active low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCK</td>
<td>Input</td>
<td>SPI Clock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MOSI</td>
<td>Input</td>
<td>SPI Serial Input</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MISO</td>
<td>Output</td>
<td>SPI Serial Output</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IRQ</td>
<td>Output</td>
<td>Interrupt, active low</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7: Pin functions of the nRF24L01

Standby Modes
Standby-I mode is used to minimize average current consumption while maintaining short start up times. In this mode, part of the crystal oscillator is active. In Standby-II mode some clock buffers are active. Standby-II occurs when CE is held high on a PTX device with empty TX FIFO. The configuration word content is maintained during Standby modes. SPI interface may be activated. For start up time see Table 13.

Power Down Mode
In power down nRF24L01 is disabled with minimal current consumption. When entering this mode the device is not active, but all registers values available from the SPI interface are maintained during power down and the SPI interface may be activated (CSN=0). For start up time see Table 13. The power down is controlled by the PWR_UP bit in the CONFIG register.
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

Packet Handling Methods
nRF24L01 has the following Packet Handling Methods:
- ShockBurst™ (compatible with nRF2401, nRF24E1, nRF2402 and nRF24E2 with 1Mbps data rate, see page 24)
- Enhanced ShockBurst™

ShockBurst™
ShockBurst™ makes it possible to use the high data rate offered by nRF24L01 without the need of a costly, high-speed microcontroller (MCU) for data processing/clock recovery. By placing all high speed signal processing related to RF protocol on-chip, nRF24L01 offers the application microcontroller a simple SPI compatible interface, the data rate is decided by the interface-speed the microcontroller itself sets up. By allowing the digital part of the application to run at low speed, while maximizing the data rate on the RF link, ShockBurst™ reduces the average current consumption in applications.
In ShockBurst™ RX, IRQ notifies the MCU when a valid address and payload is received respectively. The MCU can then clock out the received payload from an nRF24L01 RX FIFO.
In ShockBurst™ TX, nRF24L01 automatically generates preamble and CRC, see Table 12. IRQ notifies the MCU that the transmission is completed. All together, this means reduced memory demand in the MCU resulting in a low cost MCU, as well as reduced software development time. nRF24L01 has a three level deep RX FIFO (shared between 6 pipes) and a three level deep TX FIFO. The MCU can access the FIFOs at any time, in power down mode, in standby modes, and during RF packet transmission. This allows the slowest possible SPI interface compared to the average datarate, and may enable usage of an MCU without hardware SPI.

Enhanced ShockBurst™
Enhanced ShockBurst™ is a packet handling method with functionality that makes bi-directional link protocol implementation easier and more efficient. In a typical bi-directional link, one will let the terminating part acknowledge received packets from the originating part in order to make it possible to detect data loss. Data loss can then be recovered by retransmission. The idea with Enhanced ShockBurst™ is to let nRF24L01 handle both acknowledgement of received packets and retransmissions of lost packets, without involvement from the microcontroller.
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

Figure 4: nRF24L01 in a star network configuration

An nRF24L01 configured as primary RX (PRX) will be able to receive data through 6 different data pipes, see Figure 4. A data pipe will have a unique address but share the same frequency channel. This means that up to 6 different nRF24L01 configured as primary TX (PTX) can communicate with one nRF24L01 configured as PRX, and the nRF24L01 configured as PRX will be able to distinguish between them. Data pipe 0 has a unique 40 bit configurable address. Each of data pipe 1-5 has an 8 bit unique address and shares the 32 most significant address bits. All data pipes can perform full Enhanced ShockBurst™ functionality. nRF24L01 will use the data pipe address when acknowledging a received packet. This means that nRF24L01 will transmit ACK with the same address as it receives payload at. In the PTX device data pipe 0 is used to received the acknowledge, and therefore the receive address for data pipe 0 has to equal to the transmit address to be able to receive the acknowledge. See Figure 5 for addressing example.
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

Figure 5: Example on how the acknowledgement addressing is done

An nRF24L01 configured as PTX with Enhanced ShockBurst™ enabled, will use the ShockBurst™ feature to send a packet whenever the microcontroller wants to. After the packet has been transmitted, nRF24L01 will switch on its receiver and expect an acknowledgement to arrive from the terminating part. If this acknowledgement fails to arrive, nRF24L01 will retransmit the same packet until it receives an acknowledgement or the number of retries exceeds the number of allowed retries given in the SETUP_RETR_ARC register. If the number of retries exceeds the number of allowed retries, this will show in the STATUS register bit MAX_RT and gives an interrupt.

Whenever an acknowledgement is received by an nRF24L01 it will consider the last transmitted packet as delivered. It will then be cleared from the TX FIFO, and the TX_DS IRQ source will be set high.

With Enhanced ShockBurst™ nRF24L01 offers the following benefits:

- Highly reduced current consumption due to short time on air and sharp timing when operating with acknowledgement traffic
- Lower system cost. Since the nRF24L01 handles all the high-speed link layer operations, like re-transmission of lost packet and generating acknowledgement to received packets, it is no need for hardware SPI on the system microcontroller to interface the nRF24L01. The interface can be done by using general purpose IO pins on a low cost microcontroller where the SPI is emulated in firmware. With the nRF24L01 this will be sufficient speed even when running a bi-directional link.
- Greatly reduced risk of “on-air” collisions due to short time on air
- Easier firmware development since the link layer is integrated on chip
PRELIMINARY PRODUCT SPECIFICATION
nRF24L01 Single Chip 2.4 GHz Radio Transceiver

Enhanced ShockBurst™ Transmitting Payload:
1. The configuration bit PRIM_RX has to be low.
2. When the application MCU has data to send, the address for receiving node (TX_ADDR) and payload data (TX_PLD) has to be cycled into nRF24L01 via the SPI interface. The width of TX-payload is counted from number of bytes written into the TX FIFO from the MCU. TX_PLD must be written continuously while holding CSN low. TX_ADDR does not have to be rewritten if it is unchanged from last transmit. If the PTX device shall receive acknowledge, data pipe 0 has to be configured to receive the acknowledge. The receive address for data pipe 0 (RX_ADDR_P0) has to be equal to the transmit address (TX_ADDR) in the PTX device. For the example in Figure 5 the following address settings have to be performed for the TX5 device and the RX device:
   TX5 device: TX_ADDR = 0xB3B4B5B605
   TX5 device: RX_ADDR_P0 = 0xB3B4B5B605
   RX device: RX_ADDR_P5 = 0xB3B4B5B605
3. A high pulse on CE starts the transmission. The minimum pulse width on CE is 10 µs.
4. nRF24L01 ShockBurst™:
   - Radio is powered up
   - 16 MHz internal clock is started.
   - RF packet is completed (see the packet description)
   - Data is transmitted at high speed (1 Mbps or 2 Mbps configured by MCU).
5. If auto acknowledgement is activated (Auto retransmit counter not equal zero, ENAA_P0=1) the radio goes into RX mode immediately. If a valid packet has been received in the valid acknowledgement time window, the transmission is considered a success. The TX_DS bit in the status register is set high and the payload is removed from TX FIFO. If a valid acknowledgement is not received in the specified time window, the payload is resent (if auto retransmit is enabled). If the auto retransmit counter (ARC_CNT) exceeds the programmed maximum limit (ARC), the MAX_RT bit in the status register is set high. The payload in TX FIFO is NOT removed. The IRQ pin will be active when MAX_RT or TX_DS is high. To turn off the IRQ pin, the interrupt source must be reset by writing to the status register (see Interrupt chapter). If no acknowledgement is received for a packet after the maximum number of retries, no further packets can be sent before the MAX_RX interrupt is cleared. The packet loss counter (PLOS_CNT) is incremented at each MAX_RT interrupt. i.e. ARC_CNT counts the number of retries that was required to get a single packet through. PLOS_CNT counts the number of packets that did not get through even after maximum number of retries.
6. The device goes into Standby-I mode if CE is low. Otherwise next payload in TX FIFO will be sent. If TX FIFO is empty and CE is still high, the device will enter Standby-II mode.
7. If the device is in Standby-II mode, it will go to Standby-I mode immediately if CE is set low.

Enhanced ShockBurst™ Receive Payload:
1. RX is selected by setting the PRIM_RX bit in the configuration register to high, and then setting CE high.
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

2. After 130μs nRF24L01 is monitoring the air for incoming communication.
3. When a valid packet has been received (matching address and correct CRC), the payload is stored in the RX-FIFO, and the RX_DR bit in status register is set high. The IRQ pin will be active when RX_DR is high. RX_P_NO in status register will indicate what data pipe the payload has been received in.
4. If auto acknowledgement is enabled, an acknowledgement is sent back.
5. MCU sets the CE pin low to enter Standby-1 mode (low current mode).
6. MCU can clock out the payload data at a suitable rate via the SPI interface.
7. The device is now ready for entering TX or RX mode or power down mode.

Auto Acknowledgement (RX)
The auto acknowledgement function reduces the load of the external microcontroller, and may remove the need for dedicated SPI hardware in a mouse keyboard or comparable systems, and hence reduce cost and average current consumption. Auto acknowledgement can be configured individually for each data pipe via the SPI interface.

If auto acknowledgement is enabled and a valid packet (correct data pipe address and CRC) is received, the device will enter TX mode and send an acknowledgement packet. After the device has sent the acknowledgement packet, normal operation resumes, and the mode is determined by the PRIM_RX register and CE pin.

Auto Re-Transmission (ART) (TX)
An auto retransmission function is available. It will be used at the TX side in an auto acknowledgement system. In the SETUP_RETR register it will be possible to state how many times the data in the data register will be resent if data is not acknowledged. After each sending, the device will enter RX mode and wait a specified time period for acknowledgement. When the acknowledgement packet is received, the device will return to normal transmit function. If there is no more unused data in the TX FIFO and the CE pin is low, the device will go into Standby-1 mode. If the acknowledgement is not received, the device will go back to TX mode and resend the data. This will continue until acknowledgement is received, or a time out occurs (i.e. the maximum number of sending is reached). The only way to reset this is to set the PWR_UP bit low or let the auto retransmission finish. A packet loss counter will be incremented each time a packet does not succeed to reach the destination before time out. (Time out is indicated by the MAX_RT interrupt.) The packet loss counter is reset when writing to the RF channel register.

Packet Identity (PID) and CRC used by Enhanced ShockBurst™
Each packet contains a two bit wide PID field to detect if the received packet is new or resent. The PID will prevent that the PRX device presents the same payload more than once to the microcontroller. This PID field is incremented at the TX side for each new packet received via the SPI interface. The PID and CRC field is used by the PRX device to determine whether a packet is resent or new. When several data is lost on the link, the PID fields may in some cases become equal to last received PID. If a packet has the same PID as the previous packet, nRF24L01 will compare the CRC sums from both packets. If they also are equal, the last received packet is considered as a copy of the previous and is discarded.
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

1: PRX device:
The PRX device compares the received PID with the last PID. If the PID fields are different, the packet is considered to be new. If the PID is equal to last received PID, the received packet might be the same as last time. The receiver must check if the CRC is equal to the previous CRC. If the CRC is equal to the previous one, the packet is probably the same, and will be discarded.

2: PTX device:
The transmitter increments the PID field each time it sends a new packet.

Figure 6 PID generation/detection

The length of the CRC is configurable through the SPI interface. It is important to notice that the CRC is calculated over the whole packet including address, PID and payload. No packet is accepted as correct if the CRC fails. This is an extra requirement for packet acceptance that is not illustrated in the figure above.

Stationary Disturbance Detection – CD
Carrier Detect (CD) is set high when an in-band RF signal is detected in RX mode, otherwise CD is low. The internal CD signal is filtered before presented to CD register. The internal CD signal must be high for at least 128μs.

In Enhanced ShockBurst™ it is recommended to use the Carrier Detect functionality only when the PTX device does not succeed to get packets through, as indicated by the MAX RT interrupt for single packets and by the packet loss counter (PLOS_CNT) if several packets are lost. If the PLOS_CNT in the PTX device indicates too high rate of packet losses, the device can be configured to a PRX device for a short time \((T_{total} + CD\text{-}filter\ delay = 130\mu s + 128\mu s = 258\mu s)\) to check CD. If CD was high (jam situation), the frequency channel should be changed. If CD was low
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

(out of range), it may continue on the same frequency channel, but perform other adjustments. (A dummy write to the RF.CH will clear the PLOS_CNT.)

Data Pipes
nRF24L01 configured as PRX can receive data addressed to 6 different data pipes in one physical frequency channel. Each data pipe has its own unique address and can be configured to have individual behavior.

The data pipes are enabled with the bits in the EN_RXADDR register. By default only data pipe 0 and 1 are enabled.
The address for each data pipe is configured in the RX_ADDR_Px registers. Always ensure that none of the data pipes have the exact same address.

Data pipe 0 has a unique 40 bit configurable address. Data pipes 1-5 share the 32 most significant address bits and have only the L3Byte unique for each data pipe. Figure 7 shows an example of how data pipes 0-5 are addressed. All pipes can have up to 40 bit address, but for pipe 1-5 only the L3Byte is different.

<table>
<thead>
<tr>
<th>Data pipe 0 (RX_ADDR_P0)</th>
<th>0x07</th>
<th>0xD3</th>
<th>0xF0</th>
<th>0x35</th>
<th>0x77</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data pipe 1 (RX_ADDR_P1)</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
</tr>
<tr>
<td>Data pipe 2 (RX_ADDR_P2)</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC3</td>
</tr>
<tr>
<td>Data pipe 3 (RX_ADDR_P3)</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC4</td>
</tr>
<tr>
<td>Data pipe 4 (RX_ADDR_P4)</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC5</td>
</tr>
<tr>
<td>Data pipe 5 (RX_ADDR_P5)</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC2</td>
<td>0xC6</td>
</tr>
</tbody>
</table>

Figure 7: Addressing data pipes 0-5

When a packet has been received at one of the data pipes and the data pipe is setup to generate acknowledgement, nRF24L01 will generate an acknowledgement with an address that equals the data pipe address where the packet was received.

Some configuration settings are common to all data pipes and some are individual. The following settings are common to all data pipes:
- CRC enabled/disabled (CRC always enabled when ESB is enabled)
- CRC encoding scheme
- RX address width
- Frequency channel
- RF data rate
- LNA gain
- RF output power
PRELIMINARY PRODUCT SPECIFICATION
nRF24L01 Single Chip 2.4 GHz Radio Transceiver

DEVICE CONFIGURATION
All configuration of nRF24L01 is defined by values in some configuration registers. All these registers are writable via the SPI interface.

SPI Interface
The SPI interface is a standard SPI interface with a maximum data rate of 10Mbps. Most registers are readable.

SPI Instruction Set
The available commands to be used on the SPI interface are shown below. Whenever CSN is set low the interface expects an instruction. Every new instruction must be started by a high to low transition on CSN.

In parallel to the SPI instruction word applied on the MOSI pin, the STATUS register is shifted serially out on the MISO pin.

The serial shifting SPI commands is on the format:
<Instruction word: MSBit to LSBit (one byte)>
<Data bytes: LSByte to MSByte, MSBit in each byte first>
See Figure 8 and Figure 9.

<table>
<thead>
<tr>
<th>Instruction Name</th>
<th>Instruction Format [binary]</th>
<th># Data Bytes</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_REGISTER</td>
<td>000A AAAAA</td>
<td>1 to 5 LSByte first</td>
<td>Read registers. AAAAAA = 5 bit Memory Map Address</td>
</tr>
<tr>
<td>W_REGISTER</td>
<td>001A AAAAA</td>
<td>1 to 5 LSByte first</td>
<td>Write registers. AAAAAA = 5 bit Memory Map Address Execute in power down or standby modes only.</td>
</tr>
<tr>
<td>R_RX_PAYLOAD</td>
<td>0110 0001</td>
<td>1 to 32 LSByte first</td>
<td>Read RX-payload: 1 – 32 bytes. A read operation will always start at byte 0. Payload will be deleted from FIFO after it is read. Used in RX mode.</td>
</tr>
<tr>
<td>W_TX_PAYLOAD</td>
<td>1010 0000</td>
<td>1 to 32 LSByte first</td>
<td>Used in TX mode. Write TX-payload: 1 – 32 bytes. A write operation will always start at byte 0.</td>
</tr>
<tr>
<td>FLUSH_TX</td>
<td>1110 0001</td>
<td>0</td>
<td>Flush TX FIFO, used in TX mode</td>
</tr>
<tr>
<td>FLUSH_RX</td>
<td>1110 0010</td>
<td>0</td>
<td>Flush RX FIFO, used in RX mode Should not be executed during transmission of acknowledge, i.e. acknowledge package will not be completed.</td>
</tr>
<tr>
<td>REUSE_TX_PL</td>
<td>1110 0011</td>
<td>0</td>
<td>Used for a PTX device Reuse last sent payload. Packets will be repeatedly resent as long as CE is high. TX payload reuse is active until W_TX_PAYLOAD or FLUSH TX is executed. TX payload reuse must not be activated or deactivated during package transmission</td>
</tr>
<tr>
<td>NOP</td>
<td>1111 1111</td>
<td>0</td>
<td>No Operation. Might be used to read the STATUS register</td>
</tr>
</tbody>
</table>

Table 8 Instruction set for the nRF24L01 SPI interface.

The W_REGISTER and R_REGISTER may operate on single or multi-byte registers. When accessing multi-byte registers one will read or write MSBit of LSByte first. The
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

writing can be terminated before all bytes in a multi-byte register has been written. In this case the unwritten MSByte(s) will remain unchanged. E.g. the LSByte of RX_ADDR_P0 can be modified by writing only one byte to the RX_ADDR_P0 register. The content of the status register will always be read to MISO after a high to low transition on CSN.

Interrupt

The nRF24L01 has an active low interrupt pin (IRQ). The interrupt pin is activated when TX_DS, RX_DR or MAX_RT is set high in status register. When MCU writes '1' to the interrupt source, the IRQ pin will go inactive. The interrupt mask part of the CONFIG register is used to mask out the interrupt sources that are allowed to set the IRQ pin low. By setting one of the MASK bits high, the corresponding interrupt source will be disabled. By default all interrupt sources are enabled.

SPI Timing

The interface supports SPI. SPI operation and timing is given in Figure 8 to Figure 10 and in Table 9 and Table 10. The device must be in one of the standby modes or power down mode before writing to the configuration registers. In Figure 8 to Figure 10 the following notations are used:

Cn – SPI Instruction Bit
Sn – Status Register Bit
Dn – Data Bit (note: LSByte to MSByte, MSBit in each byte first)

![Figure 8 SPI read operation.](image)

![Figure 9 SPI write operation.](image)
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

Figure 10 SPI NOP timing diagram.

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>SYMBOL</th>
<th>MIN</th>
<th>MAX</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data to SCK Setup</td>
<td>Tdc</td>
<td>2</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK to Data Hold</td>
<td>Tdh</td>
<td>2</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>CSN to Data Valid</td>
<td>Tsed</td>
<td>33</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK to Data Valid</td>
<td>Tcd</td>
<td>52</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK Low Time</td>
<td>Tcl</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK High Time</td>
<td>Tch</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK Frequency</td>
<td>Fseck</td>
<td>0</td>
<td>10</td>
<td>MHz</td>
</tr>
<tr>
<td>SCK Rise and Fall</td>
<td>Tr,Tf</td>
<td>tbd</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>CSN to SCK Setup</td>
<td>Tcc</td>
<td>2</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK to CSN Hold</td>
<td>Tech</td>
<td>2</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>CSN Inactive time</td>
<td>Tcwh</td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>CSN to Output High Z</td>
<td>Tedz</td>
<td>33</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

Table 9 SPI timing parameters (C_{load} = 5pF).

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>SYMBOL</th>
<th>MIN</th>
<th>MAX</th>
<th>UNITS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data to SCK Setup</td>
<td>Tdc</td>
<td>2</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK to Data Hold</td>
<td>Tdh</td>
<td>2</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>CSN to Data Valid</td>
<td>Tsed</td>
<td>38</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK to Data Valid</td>
<td>Tcd</td>
<td>54</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK Low Time</td>
<td>Tcl</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK High Time</td>
<td>Tch</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK Frequency</td>
<td>Fseck</td>
<td>0</td>
<td>10</td>
<td>MHz</td>
</tr>
<tr>
<td>SCK Rise and Fall</td>
<td>Tr,Tf</td>
<td>tbd</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>CSN to SCK Setup</td>
<td>Tcc</td>
<td>2</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>SCK to CSN Hold</td>
<td>Tech</td>
<td>2</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>CSN Inactive time</td>
<td>Tcwh</td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>CSN to Output High Z</td>
<td>Tedz</td>
<td>38</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

Table 10 SPI timing parameters (C_{load} = 10pF).
## MEMORY MAP

All undefined bits in the table below are redundant. They will be read out as '0'.

<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>Mnemonic</th>
<th>Bit</th>
<th>Reset Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>CONFIG</td>
<td></td>
<td>R/W</td>
<td></td>
<td>Configuration Register</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>7</td>
<td>0</td>
<td>R/W</td>
<td>Only 'W' allowed</td>
</tr>
<tr>
<td>6</td>
<td>MASK_RX_DR</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Mask interrupt caused by RX_RD  1: Interrupt not reflected on the IRQ pin 0: Reflect RX_RXDR as active low interrupt on the IRQ pin</td>
</tr>
<tr>
<td>5</td>
<td>MASK_TX_DS</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Mask interrupt caused by TX_DS  1: Interrupt not reflected on the IRQ pin 0: Reflect TX_DS as active low interrupt on the IRQ pin</td>
</tr>
<tr>
<td>4</td>
<td>MASK_MAX_RT</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Mask interrupt caused by MAX_RT 1: Interrupt not reflected on the IRQ pin 0: Reflect MAX_RT as active low interrupt on the IRQ pin</td>
</tr>
<tr>
<td>3</td>
<td>EN_CRC</td>
<td>1</td>
<td>R/W</td>
<td></td>
<td>Enable CRC. Forced high if one of the bits in the EN_AA is high</td>
</tr>
<tr>
<td>2</td>
<td>CRCO</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>CRC encoding scheme  '0' - 1 byte  '1' - 2 bytes</td>
</tr>
<tr>
<td>1</td>
<td>PWR_UP</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>1: POWER_UP, 0: POWER_DOWN</td>
</tr>
<tr>
<td>0</td>
<td>PRM_RX</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>1: PRX, 0: TX</td>
</tr>
<tr>
<td>7</td>
<td>EN_AA</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable 'Auto Acknowledgment' Function Disable this functionality to be compatible with NRF2401, see page 25</td>
</tr>
<tr>
<td>6</td>
<td>ENAA_P5</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable auto ack. data pipe 5</td>
</tr>
<tr>
<td>5</td>
<td>ENAA_P4</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable auto ack. data pipe 4</td>
</tr>
<tr>
<td>4</td>
<td>ENAA_P3</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable auto ack. data pipe 3</td>
</tr>
<tr>
<td>3</td>
<td>ENAA_P2</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable auto ack. data pipe 2</td>
</tr>
<tr>
<td>2</td>
<td>ENAA_P1</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable auto ack. data pipe 1</td>
</tr>
<tr>
<td>1</td>
<td>ENAA_P0</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable auto ack. data pipe 0</td>
</tr>
<tr>
<td>6</td>
<td>EN_RXADDR</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enabled RX Addresses</td>
</tr>
<tr>
<td>5</td>
<td>ERX_P5</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable data pipe 5</td>
</tr>
<tr>
<td>4</td>
<td>ERX_P4</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable data pipe 4</td>
</tr>
<tr>
<td>3</td>
<td>ERX_P3</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable data pipe 3</td>
</tr>
<tr>
<td>2</td>
<td>ERX_P2</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable data pipe 2</td>
</tr>
<tr>
<td>1</td>
<td>ERX_P1</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable data pipe 1</td>
</tr>
<tr>
<td>0</td>
<td>ERX_P0</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Enable data pipe 0</td>
</tr>
<tr>
<td>2</td>
<td>SETUP_AW</td>
<td>1</td>
<td>R/W</td>
<td></td>
<td>Setup of Address Widths (common for all data pipes)</td>
</tr>
<tr>
<td>2</td>
<td>AW</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>RX/TX Address field width '00' - Illegal  '01' - 3 bytes  '10' - 4 bytes  '11' - 5 bytes LSBYTE will be used if address width below 5 bytes</td>
</tr>
<tr>
<td>1</td>
<td>SETUP_RETR</td>
<td>1</td>
<td>R/W</td>
<td></td>
<td>Setup of Automatic Retransmission</td>
</tr>
<tr>
<td>1</td>
<td>ARD</td>
<td>0</td>
<td>R/W</td>
<td></td>
<td>Auto Re-transmit Delay</td>
</tr>
</tbody>
</table>
# PRELIMINARY PRODUCT SPECIFICATION

**nRF24L01 Single Chip 2.4 GHz Radio Transceiver**

<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>Mnemonic</th>
<th>Bit</th>
<th>Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ARC</td>
<td>3:0</td>
<td>0011</td>
<td>R/W</td>
<td>Auto Retransmit Count</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'0000' – Wait 250+86uS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'0001' – Wait 500+86uS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'0010' – Wait 750+86uS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'1111' – Wait 4000+86uS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(Delay defined from end of transmission to start of next transmission)</td>
</tr>
</tbody>
</table>

| 05            | RF.CH    | 6:0 | 00000010 | R/W | Sets the frequency channel nRF24L01 operates on |

<table>
<thead>
<tr>
<th>06</th>
<th>RF_SETUP</th>
<th>7:5</th>
<th>000</th>
<th>R/W</th>
<th>RF Setup Register</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PLL_LOCK</td>
<td>4</td>
<td>0</td>
<td>R/W</td>
<td>Force PLL lock signal</td>
</tr>
<tr>
<td></td>
<td>RF_DR</td>
<td>3</td>
<td>1</td>
<td>R/W</td>
<td>Data Rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'0' – 1 Mbps</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'1' – 2 Mbps</td>
</tr>
<tr>
<td></td>
<td>RF_PWR</td>
<td>2:1</td>
<td>11</td>
<td>R/W</td>
<td>Set RF output power in TX mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'00' – -18 dBm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'01' – -12 dBm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'0' – 6 dBm</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>'1' – 0 dBm</td>
</tr>
<tr>
<td></td>
<td>LNA_HCURR</td>
<td>0</td>
<td>1</td>
<td>R/W</td>
<td>Setup LNA gain</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>07</th>
<th>STATUS</th>
<th></th>
<th></th>
<th></th>
<th>Status Register (In parallel to the SPI instruction word applied on the MOSI pin, the STATUS register is shifted serially out on the MISO pin)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reserved</td>
<td>7</td>
<td>0</td>
<td>R/W</td>
<td>Only '0' allowed</td>
</tr>
<tr>
<td></td>
<td>RX_DR</td>
<td>6</td>
<td>0</td>
<td>R/W</td>
<td>Data Ready RX FIFO interrupt. Set high when new data arrives RX FIFO. Write 1 to clear bit.</td>
</tr>
<tr>
<td></td>
<td>TX_DS</td>
<td>5</td>
<td>0</td>
<td>R/W</td>
<td>Data Sent TX FIFO interrupt. Set high when packet sent on TX. If AUTO_ACK is activated, this bit will be set high only when ACK is received. Write 1 to clear bit.</td>
</tr>
<tr>
<td></td>
<td>MAX_RT</td>
<td>4</td>
<td>0</td>
<td>R/W</td>
<td>Maximum number of TX retries interrupt. Write 1 to clear bit. If MAX_RT is set, it must be cleared to enable further communication.</td>
</tr>
<tr>
<td></td>
<td>RX_P_NO</td>
<td>3:1</td>
<td>111</td>
<td>R</td>
<td>Data pipe number for the payload available for reading from RX_FIFO.</td>
</tr>
<tr>
<td></td>
<td>TX_FULL</td>
<td>0</td>
<td>0</td>
<td>R</td>
<td>TX FIFO full flag. 1: TX FIFO full. 0:</td>
</tr>
</tbody>
</table>

---

13 The Data Ready interrupt is set by a new packet arrival event. The procedure for handling this interrupt should be: 1) read payload via SPI, 2) clear RX_DR interrupt, 3) read FIFO STATUS to check if there are more payloads available in RX_FIFO, 4) if there are more data in RX_FIFO, repeat from 1).
### PRELIMINARY PRODUCT SPECIFICATION

**nRF24L01 Single Chip 2.4 GHz Radio Transceiver**

<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>Mnemonic</th>
<th>Bit</th>
<th>Reset Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>08</td>
<td>OBSERVE_TX</td>
<td></td>
<td></td>
<td></td>
<td>Transient observe register</td>
</tr>
<tr>
<td></td>
<td>PLOS_CNT</td>
<td>7:4</td>
<td>0</td>
<td>R</td>
<td>Packet Loss Counter. The register is reset by writing to RF_CH. The counter restarts after 15 lost packets. See page 14 and 16.</td>
</tr>
<tr>
<td></td>
<td>ARC_CNT</td>
<td>3:0</td>
<td>0</td>
<td>R</td>
<td>Current value on reset counter. The counter is reset when transmission of a new packet starts. See page 14.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>09</th>
<th>CD</th>
<th></th>
<th></th>
<th></th>
<th>Decimal. Reserved for later use. Default is 0.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CD</td>
<td>7:1</td>
<td>000000</td>
<td>R</td>
<td>Carrier Detect. See page 16.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0A</th>
<th>RX_ADDR_P0</th>
<th>39:0</th>
<th>0xE7E7E7E7E7</th>
<th>R/W</th>
<th>Receive address data page 0. 5 Bytes maximum length. (LSByte is written first)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0B</td>
<td>RX_ADDR_P1</td>
<td>39:0</td>
<td>0xC2C2C2C2C2</td>
<td>R/W</td>
<td>Receive address data page 1. 5 Bytes maximum length. (LSByte is written first)</td>
</tr>
<tr>
<td>0C</td>
<td>RX_ADDR_P2</td>
<td>7:0</td>
<td>0xC3</td>
<td>R/W</td>
<td>Receive address data page 2. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]</td>
</tr>
<tr>
<td>0D</td>
<td>RX_ADDR_P3</td>
<td>7:0</td>
<td>0xC4</td>
<td>R/W</td>
<td>Receive address data page 3. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]</td>
</tr>
<tr>
<td>0E</td>
<td>RX_ADDR_P4</td>
<td>7:0</td>
<td>0xC5</td>
<td>R/W</td>
<td>Receive address data page 4. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]</td>
</tr>
<tr>
<td>0F</td>
<td>RX_ADDR_P5</td>
<td>7:0</td>
<td>0xC6</td>
<td>R/W</td>
<td>Receive address data page 5. Only LSB. MSBytes will be equal to RX_ADDR_P1[39:8]</td>
</tr>
</tbody>
</table>

| 10            | TX_ADDR      | 39:0| 0xE7E7E7E7E7| R/W  | Transmit address. Used for a PTX device only. (LSByte is written first) Set RX_ADDR_P0 equal to this address to handle automatic acknowledge if this is a PTX device with Enhanced ShockBurst™ enabled. See page 14. |

<table>
<thead>
<tr>
<th>11</th>
<th>RX_PW_P0</th>
<th>7:6</th>
<th>00</th>
<th>R/W</th>
<th>Only 00 allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RX_PW_P0</td>
<td>5:0</td>
<td>0</td>
<td>R/W</td>
<td>Number of bytes in RX payload in data pipe 0 (1 to 32 bytes). 0 Not Legal 1 ~ 1 byte 32 = 32 bytes</td>
</tr>
<tr>
<td></td>
<td>RX_PW_P1</td>
<td>7:6</td>
<td>00</td>
<td>R/W</td>
<td>Only 00 allowed</td>
</tr>
<tr>
<td></td>
<td>RX_PW_P1</td>
<td>5:0</td>
<td>0</td>
<td>R/W</td>
<td>Number of bytes in RX payload in data pipe 1 (1 to 32 bytes). 0 Not Legal 1 ~ 1 byte 32 = 32 bytes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>13</th>
<th>RX_PW_P2</th>
<th>7:6</th>
<th>00</th>
<th>R/W</th>
<th>Only 00 allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RX_PW_P2</td>
<td>5:0</td>
<td>0</td>
<td>R/W</td>
<td>Number of bytes in RX payload in data pipe 2 (1 to 32 bytes).</td>
</tr>
</tbody>
</table>
## PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

<table>
<thead>
<tr>
<th>Address (Hex)</th>
<th>Mnemonic</th>
<th>Bit</th>
<th>Reset Value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>RX_PW_P3</td>
<td>7:6</td>
<td>00</td>
<td>R/W</td>
<td>Only '0' allowed</td>
</tr>
<tr>
<td></td>
<td>RX_PW_P3</td>
<td>5:0</td>
<td>0</td>
<td>R/W</td>
<td>Number of bytes in RX payload in data pipe 3 (1 to 32 bytes).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 Not Legal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = 1 byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32 = 32 bytes</td>
</tr>
<tr>
<td>15</td>
<td>RX_PW_P4</td>
<td>7:6</td>
<td>00</td>
<td>R/W</td>
<td>Only '0' allowed</td>
</tr>
<tr>
<td></td>
<td>RX_PW_P4</td>
<td>5:0</td>
<td>0</td>
<td>R/W</td>
<td>Number of bytes in RX payload in data pipe 4 (1 to 32 bytes).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 Not Legal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = 1 byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32 = 32 bytes</td>
</tr>
<tr>
<td>16</td>
<td>RX_PW_P5</td>
<td>7:6</td>
<td>00</td>
<td>R/W</td>
<td>Only '0' allowed</td>
</tr>
<tr>
<td></td>
<td>RX_PW_P5</td>
<td>5:0</td>
<td>0</td>
<td>R/W</td>
<td>Number of bytes in RX payload in data pipe 5 (1 to 32 bytes).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 Not Legal</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 = 1 byte</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32 = 32 bytes</td>
</tr>
<tr>
<td>17</td>
<td>FIFO_STATUS</td>
<td>7</td>
<td>0</td>
<td>R/W</td>
<td>Only '0' allowed</td>
</tr>
<tr>
<td></td>
<td>TX_REUSE</td>
<td>6</td>
<td>0</td>
<td>R</td>
<td>Reuse last sent data packet if set high. The packet will be repeatedly retransmit as long as CE is high. TX_REUSE is set by the SPI instruction REUSE_TX_PL, and is reset by the SPI instructions W_TX_PAYLOAD or FLUSH TX</td>
</tr>
<tr>
<td></td>
<td>TX_FULL</td>
<td>5</td>
<td>0</td>
<td>R</td>
<td>TX FIFO full flag. 1: TX FIFO full. 0: Available locations in TX FIFO.</td>
</tr>
<tr>
<td></td>
<td>TX.Empty</td>
<td>4</td>
<td>1</td>
<td>R</td>
<td>TX FIFO empty flag. 1: TX FIFO empty. 0: Data in TX FIFO.</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>3:2</td>
<td>00</td>
<td>R/W</td>
<td>Only '0' allowed</td>
</tr>
<tr>
<td></td>
<td>RX_FULL</td>
<td>1</td>
<td>0</td>
<td>R</td>
<td>RX FIFO full flag. 1: RX FIFO full. 0: Available locations in RX FIFO.</td>
</tr>
<tr>
<td></td>
<td>RX.Empty</td>
<td>0</td>
<td>1</td>
<td>R</td>
<td>RX FIFO empty flag. 1: RX FIFO empty. 0: Data in RX FIFO.</td>
</tr>
<tr>
<td>N/A</td>
<td>TX.PLD</td>
<td>255:0</td>
<td>X</td>
<td>W</td>
<td>Written by separate SPI command TX data payload register 1 - 32 bytes. This register is implemented as a FIFO with 3 levels. Used in TX mode only</td>
</tr>
<tr>
<td>N/A</td>
<td>RX.PLD</td>
<td>255:0</td>
<td>X</td>
<td>R</td>
<td>Written by separate SPI command RX data payload register 1 - 32 bytes. This register is implemented as a FIFO with 3 levels. All receive channels share the same FIFO</td>
</tr>
</tbody>
</table>

Table 11 Memory map of nRF24L01
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

Configuration for compatibility with nRF24XX

How to setup nRF24L01 to receive from an nRF2401/nRF2402/nRF24E1/nRF24E2:

- Use same CRC configuration as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
- Set the PRIM_RX bit to 1
- Disable auto acknowledgement on the data pipe that will be addressed
- Use the same address width as the PTX device
- Use the same frequency channel as the PTX device
- Select data rate 1Mbit/s on both nRF24L01 and nRF2401/nRF2402/nRF24E1/nRF24E2
- Set correct payload width on the data pipe that will be addressed
- Set PWR_UP and CE high

How to setup nRF24L01 to transmit to an nRF2401/nRF24E1:

- Use same CRC configuration as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
- Set the PRIM_RX bit to 0
- Set the Auto Retransmit Count to 0 to disable the auto retransmit functionality
- Use the same address width as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
- Use the same frequency channel as the nRF2401/nRF2402/nRF24E1/nRF24E2 uses
- Select data rate 1Mbit/s on both nRF24L01 and nRF2401/nRF2402/nRF24E1/nRF24E2
- Set PWR_UP high
- Clock in a payload that has the same length as the nRF2401/nRF2402/nRF24E1/nRF24E2 is configured to receive
- Pulse CE to send the packet
PRELIMINARY PRODUCT SPECIFICATION
nRF24L01 Single Chip 2.4 GHz Radio Transceiver

PACKET DESCRIPTION
An Enhanced ShockBurst™ packet with payload (1-32 bytes).

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Address 3-5 byte</th>
<th>9 bit</th>
<th>Payload 1 - 32 byte</th>
<th>CRC 0/1/2 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A ShockBurst™ packet compatible to nRF2401/nRF2402/nRF24E1/nRF24E2 devices.

<table>
<thead>
<tr>
<th>Preamble</th>
<th>Address 3-5 byte</th>
<th>Payload 1 - 32 byte</th>
<th>CRC 0/1/2 byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 byte</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Preamble is used to detect 0 and 1 levels. It is stripped off (RX) and added (TX) by nRF24L01. The preamble is 8 bits long.
- The address field contains the receiver address.
- The address can be 3, 4, or 5 bytes wide.
- The address fields can be individually configured for all RX channels and the TX channel.
- Address is automatically removed from received packets.\(^{14}\)

- Flags
- PID: Packet Identification. 2 bits that is incremented for each new payload.
- 7 bits reserved for packet compatibility with future products.
- Not used when compatible to nRF2401

- Payload 1 - 32 bytes wide.

| CRC       | The CRC is optional. 0-2 bytes CRC
|-----------|-----------------------------------|
| The polynomial for 8 bits CRC check is \(X^8 + X^2 + X + 1\)
| The polynomial for 16 bits CRC check is \(X^{16} + X^{12} + X^3 + 1\) |

Table 12 Data packet description

\(^{14}\) Suggested use of addresses. In general more bits in the address gives less false detection, which in the end may give lower data Packet-Error-Rate (PER).

A. The address made by (5, 4, or 3) equal bytes are not recommended because it in general will make the packet-error-rate increase.
B. Addresses where the level shift only one time (i.e. 000FFFFF) could often be detected in noise that may give a false detection, which again may give raised packet-error-rate.
C. Addresses as a continuation of the preamble (hi-low toggling) will raise the Packet-Error-Rate (PER).
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

IMPORTANT TIMING DATA

The following timing applies for operation of nRF24L01.

nRF24L01 Timing Information

<table>
<thead>
<tr>
<th>nRF24L01 timing</th>
<th>Max.</th>
<th>Min.</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Down → Standby mode</td>
<td>1.5ms</td>
<td></td>
<td>Tpd2stby</td>
</tr>
<tr>
<td>Standby modes → TX/RX mode</td>
<td>130µs</td>
<td></td>
<td>Tstby2a</td>
</tr>
<tr>
<td>Minimum CE high</td>
<td></td>
<td>10µs</td>
<td>Tice</td>
</tr>
<tr>
<td>Delay from CE pos. edge to CSN low</td>
<td>4µs</td>
<td></td>
<td>Tpeec2cs</td>
</tr>
</tbody>
</table>

Table 13 Operational timing of nRF24L01

When the nRF24L01 is in power down it must always settle in Standby for 1.5ms before it can enter one of the TX or RX modes. Note that the configuration word will be lost if VDD is turned off and that the device then must be configured before going to one of the TX or RX mode.

Enhanced ShockBurst™ timing

Figure 11 Timing of Enhanced ShockBurst™ for one packet upload (2Mbps).
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

In Figure 11 the sending of one packet and the acknowledgement of this packet is shown. The loading of payload to the PTX device is not shown in the figure. The PRX device is turned into RX mode (CE=1), and the PTX device is set into TX mode (CE=1 for minimum 10 µs). After 130 µs the transmission starts and is finished after another 37 µs (1 byte payload). The transmission ends, and the PTX device is automatically turned around to RX mode to wait for the acknowledgement from the PRX device. After the PTX device has received the acknowledgement it gives an interrupt to the MCU (IRQ (TX_DS) => TX-data sent). After the PRX device has received the packet it gives an interrupt to the MCU (IRQ (RX_DR) => RX-data ready).
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

PERIPHERAL RF INFORMATION

Antenna output
The ANT1 & ANT2 output pins provide a balanced RF output to the antenna. The pins must have a DC path to VDD, either via a RF choke or via the center point in a dipole antenna. A load of 100Ω+j175Ω is recommended for maximum output power (0dBm). Lower load impedance (for instance 50Ω) can be obtained by fitting a simple matching network.

Output Power adjustment

<table>
<thead>
<tr>
<th>SPI RF_SETUP (RF_PWR)</th>
<th>RF output power</th>
<th>DC current consumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0 dBm</td>
<td>11.3 mA</td>
</tr>
<tr>
<td>10</td>
<td>-6 dBm</td>
<td>9.0 mA</td>
</tr>
<tr>
<td>01</td>
<td>-12 dBm</td>
<td>7.5 mA</td>
</tr>
<tr>
<td>00</td>
<td>-18 dBm</td>
<td>7.0 mA</td>
</tr>
</tbody>
</table>

Conditions: VDD = 3.0V, VSS = 0V, T_A = 27°C, Load impedance = 100Ω+j175Ω.

Table 14 RF output power setting for the nRF24L01.

Crystal Specification

Frequency accuracy includes initial accuracy (tolerance) and stability over temperature and aging.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>C_L</th>
<th>ESR max</th>
<th>C_max</th>
<th>Frequency accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>16MHz</td>
<td>12pF</td>
<td>100 Ω</td>
<td>7.0pF</td>
<td>±60ppm</td>
</tr>
</tbody>
</table>

Table 15 Crystal specification of the nRF24L01

To achieve a crystal oscillator solution with low power consumption and fast start-up time, it is recommended to specify the crystal with a low value of crystal load capacitance. Specifying C_L=12pF is OK, but it is possible to use up to 16pF. Specifying a lower value of crystal parallel equivalent capacitance, C_0 will also work, but this can increase the price of the crystal itself. Typically C_0=1.5pF at a crystal specified for C_max=7.0pF.

nRF24L01 sharing crystal with a micro controller.
When using a micro controller to drive the crystal reference input XC1 of the nRF24L01 transceiver some rules must be followed.

Crystal parameters:
When the micro controller drives the nRF24L01 clock input, the requirement of load capacitance C_L is set by the micro controller only. The frequency accuracy of ±60 ppm is still required to get a functional radio link. The nRF24L01 will load the crystal by 0.5pF at XC1 in addition to the PBC routing.
**Input crystal amplitude & Current consumption**

The input signal should not have amplitudes exceeding any rail voltage, but any DC-voltage within this is OK. Exceeding rail voltage will excite the ESD structure and the radio performance is degraded below specification. If testing the nRF24L01 with a RF source with no DC offset as the reference source, the input signal will go below the ground level, which is not acceptable.

![Principle of crystal oscillator](image)

*Figure 12  Principle of crystal oscillator*

The nRF24L01 crystal oscillator is amplitude regulated. To achieve low current consumption and also good signal-to-noise ratio when using an external clock, it is recommended to use an input signal larger than 0.4 V-peak. When clocked externally, XC2 is not used and can be left as an open pin.
PRELIMINARY PRODUCT SPECIFICATION

nRF24L01 Single Chip 2.4 GHz Radio Transceiver

PCB layout and de-coupling guidelines
A well-designed PCB is necessary to achieve good RF performance. Keep in mind that a poor layout may lead to loss of performance, or even functionality, if due care is not taken. A fully qualified RF-layout for the nRF24L01 and its surrounding components, including matching networks, can be downloaded from www.nordicsemi.no.

A PCB with a minimum of two layers including a ground plane is recommended for optimum performance. The nRF24L01 DC supply voltage should be de-coupled as close as possible to the VDD pins with high performance RF capacitors, see Table 16. It is preferable to mount a large surface mount capacitor (e.g. 4.7μF tantalum) in parallel with the smaller value capacitors. The nRF24L01 supply voltage should be filtered and routed separately from the supply voltages of any digital circuitry.

Long power supply lines on the PCB should be avoided. All device grounds, VDD connections and VDD bypass capacitors must be connected as close as possible to the nRF24L01 IC. For a PCB with a topside RF ground plane, the VSS pins should be connected directly to the ground plane. For a PCB with a bottom ground plane, the best technique is to have via holes as close as possible to the VSS pads. At least one via hole should be used for each VSS pin.

Full swing digital data or control signals should not be routed close to the crystal or the power supply lines.