

Romeo2 Monitor for the MC68HC908AP64 MCU

by: John Logan
8/16-bit Division
East Kilbride

Introduction

This application note describes how to use the Romeo2 Monitor program to evaluate the performance of the MC33591/2/3/4 family of RF receiver ICs (Romeo2). It shows the hardware setup required, describes how to set up the monitor, lists all available commands, and gives some examples of usage.

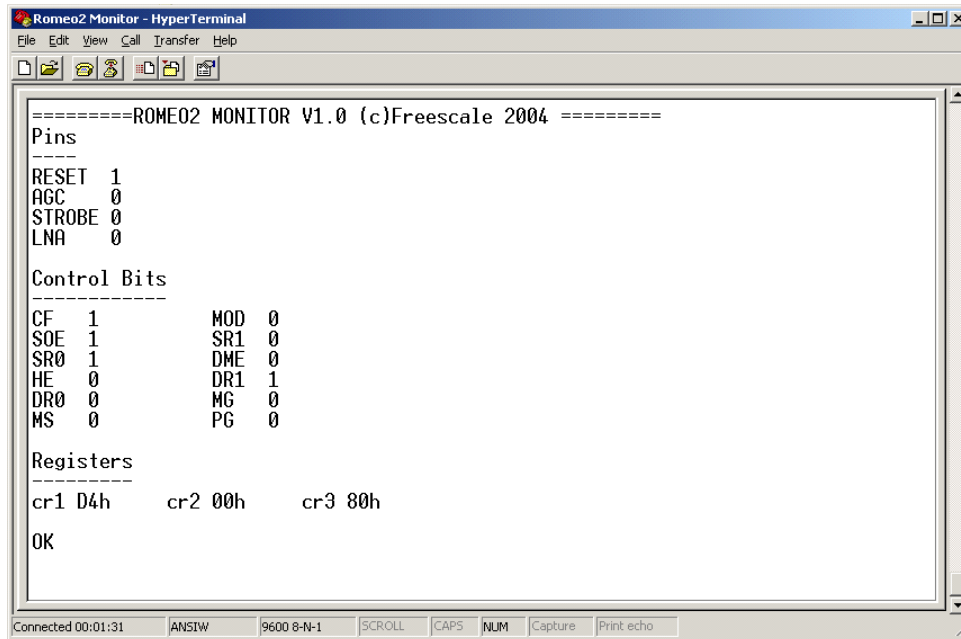
The reader should be familiar with the Romeo2 device data sheet and with the RF data formats described in the data sheet.

Description

The Romeo2 Monitor is a software program that runs on Freescale's MC68HC908AP64 Demo Board. The MC68HC908AP64 Demo Board is connected to a Romeo2 RF module and also to a PC via a serial port. The PC runs a terminal emulation program, for example, Hyperterminal. [Figure 1](#) is a screenshot of Hyperterminal showing the monitor program. [Figure 2](#) and [Figure 3](#) show the hardware setup required to use the monitor program.

Description

The user can control the pins of the Romeo2 RF module by typing simple commands into the PC. The user can also configure any of Romeo2's internal registers and enable reception of messages.



```
=====ROME02 MONITOR V1.0 (c)Freescale 2004 =====
Pins
-----
RESET 1
AGC 0
STROBE 0
LNA 0

Control Bits
-----
CF 1          MOD 0
SOE 1         SR1 0
SR0 1         DME 0
HE 0          DR1 1
DR0 0         MG 0
MS 0          PG 0

Registers
-----
cr1 D4h      cr2 00h      cr3 80h

OK

Connected 00:01:31  ANSIW  9600 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

Figure 1. Romeo2 Monitor Communicating with Hyperterminal

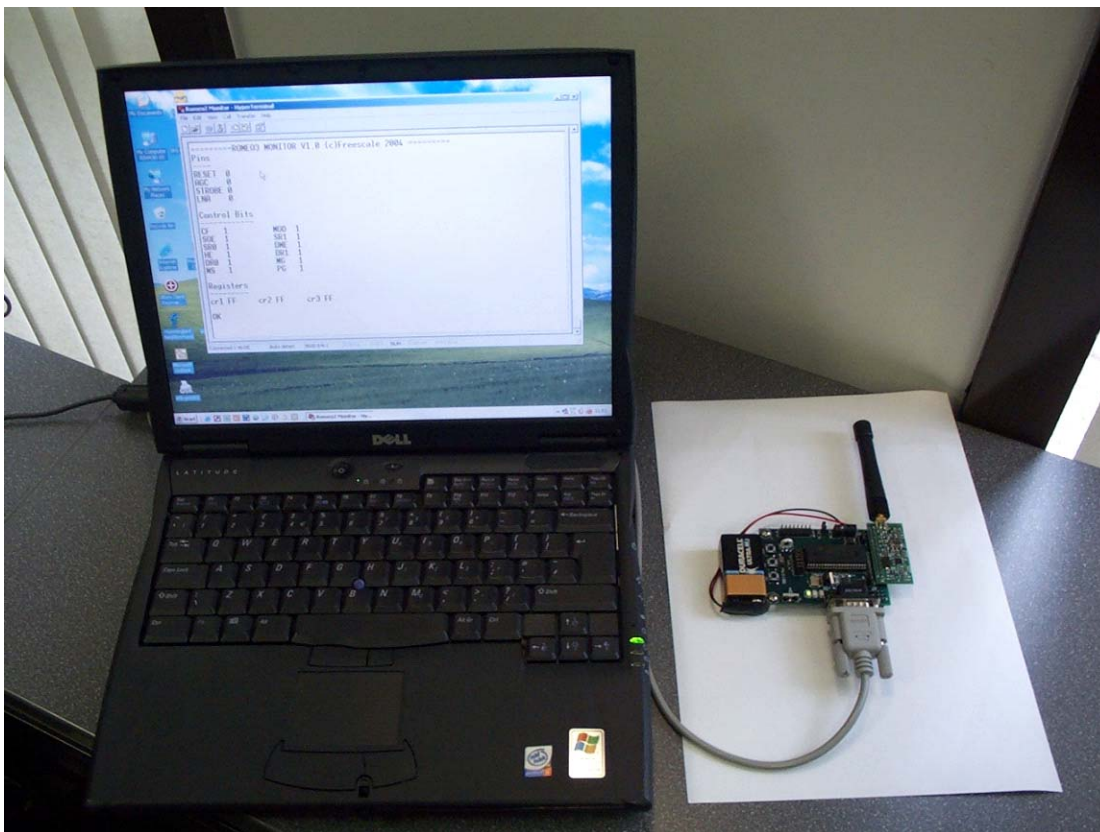


Figure 2. Romeo2 Monitor Setup

Requirements

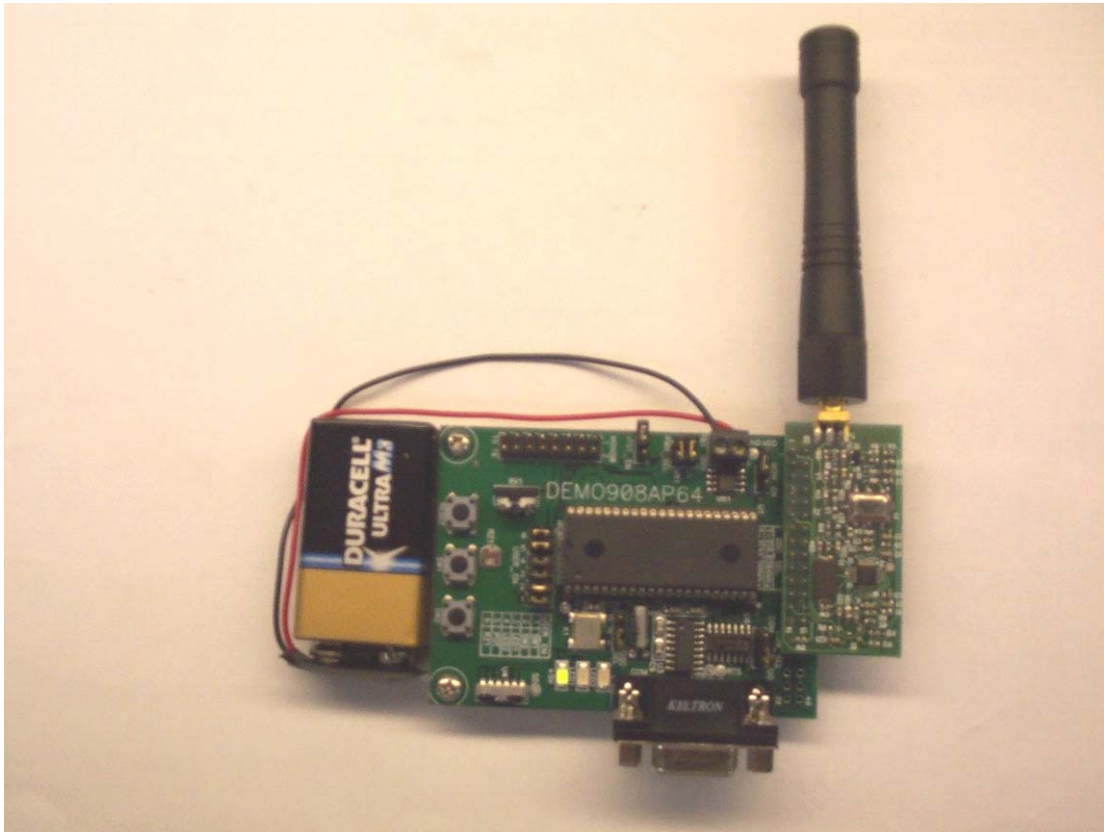


Figure 3. Close-up of MC68HC908AP64 Demo Board and Romeo2 RF Module

Requirements

To use the software, you must have the following hardware and software.

- MC68HC908AP64 Demo Board (part no. DEMO908AP64)
- Romeo2 RF module (part no MC33591MOD315 or MC33591MOD434 — others may be available)
- MetroWerks CodeWarrior version 3.0 or later
- 'AN2818SW.zip' which contains the software files
- A PC running Hyperterminal, or any other terminal program

Full source code for the Romeo2 monitor is available; this allows the monitor to be extended or modified, if required.

Programming Romeo2 Monitor into the MC68HC908AP64 MCU

If the Romeo2 monitor program has already been programmed into the MC68HC908AP64 Demo Board, the user can begin to use the monitor (see section 'Using The Software'). Otherwise, the user must follow the programming procedure described below.

1. Install power select (PWR_SEL) jumpers 1 and 2 on the MC68HC908AP64 Demo Board.
2. Install VST_EN jumper.
3. Install MON_EN jumper.
4. Select setting DEBUG on COM_SEL header.
5. Install jumper across pins 1 and 2 of header OSC_SEL.
6. Install all USER_EN jumpers.
7. Connect the serial port connector on the MC68HC908AP64 Demo Board to a PC comm. port using a 9-pin straight-through serial cable.
8. Connect a 9V power supply or battery to the power connector on the MC68HC908AP64 board.
9. Install CodeWarrior.

NOTE

You must have a copy of the CodeWarrior development tool for HC(S)08 installed. A copy of CodeWarrior is supplied with the MC68HC908AP64 Demo Board. Please follow the instructions supplied with the demo board to install CodeWarrior.

10. Unzip file Romeo2AP64MonProg.zip, which is contained within AN2818SW.zip. This unzips a CodeWarrior project containing the programming file for the monitor.

NOTE

This project does not contain any source code. If you wish to read or modify the Romeo2 monitor source code, you should unzip file Romeo2AP64MonSource.zip (contained in file AN2818SW.zip). To modify this code, you require a full license for CodeWarrior, which can be purchased from www.metrowerks.com, or your local Freescale or MetroWerks representative.

11. Start CodeWarrior (Start menu->MetroWerks->CodeWarrior->CodeWarrior IDE).
12. Select File->Open and open the file Romeo2AP64MonProg.mcp. This opens a CodeWarrior project.
13. In CodeWarrior, click on file 'Romeo2AP64MonProg' in the Target window. Then press key F5 or select Project->Debug from menu bar. This launches the debugger, which communicates with the MC68HC908AP64 Demo Board and attempts to burn the Romeo2 monitor program into FLASH memory on the MCU. The "Attempting to contact target and pass security..." window should appear. Please make sure the following options are configured correctly.
 - a. Target Hardware Type: Class 3
 - b. Serial Port: 1 (Depends on the PC COM Port)
 - c. Baud: 9600 Baud

Using the Software

- d. Target MCU Security bytes: Check the “IGNORE security failure and enter monitor mode” check box
14. Click the “Contact target with these settings...” button and follow the instructions on the screen. When the “Erase and Program Flash?” window appears, click the “Yes” button.
15. Follow the on-screen instructions for cycling the MCU power supply. The “CPROG08SZ Programmer” window should close after the MCU FLASH is programmed.
16. The monitor program has now been programmed into the FLASH memory of the MC68HC908AP64. CodeWarrior is no longer required. Shut all CodeWarrior windows and exit the program.

To use the monitor, please follow the instructions given below.

Using the Software

To use the monitor, the hardware should be set up as follows.

1. Connect the Romeo2 RF Module to connector J1 on the MC68HC908AP64 Demo Board. Pin 1 on each board must be aligned.
2. Connect an antenna to the Romeo2 RF module.

NOTE

Romeo2 RF modules are available in a range of frequencies, each is supplied with appropriate antenna.

3. Install power select (PWR_SEL) jumpers 1 and 2 on the MC68HC908AP64 Demo Board. Both jumpers must be installed.
4. Remove VST_EN jumper.
5. Remove MON_EN jumper.
6. Install jumper across pins 1 and 2 of header OSC_SEL.
7. Install jumper on COM setting of COM_SEL header.
8. Install all USER_EN jumpers.
9. Connect the serial port connector on the MC68HC908AP64 Demo Board to a PC comm. port using a 9-pin straight-through serial cable.
10. Connect 9V power supply or battery to power connector on MC68HC908AP64 board.

Refer to [Figure 3](#) for a reference setup.

The MCU board will communicate with a terminal emulation program on the PC. Hyperterminal is a common terminal program, which is supplied with the Windows operating system. It should be configured for 9600 baud, eight data bits, no parity, one stop bit, no flow control. (See [Figure 4](#).)

A setup file (Romeo2Monitor.ht) for Hyperterminal is supplied in the software package (AN2818SW.zip) for this application note and can be downloaded from www.freescale.com.

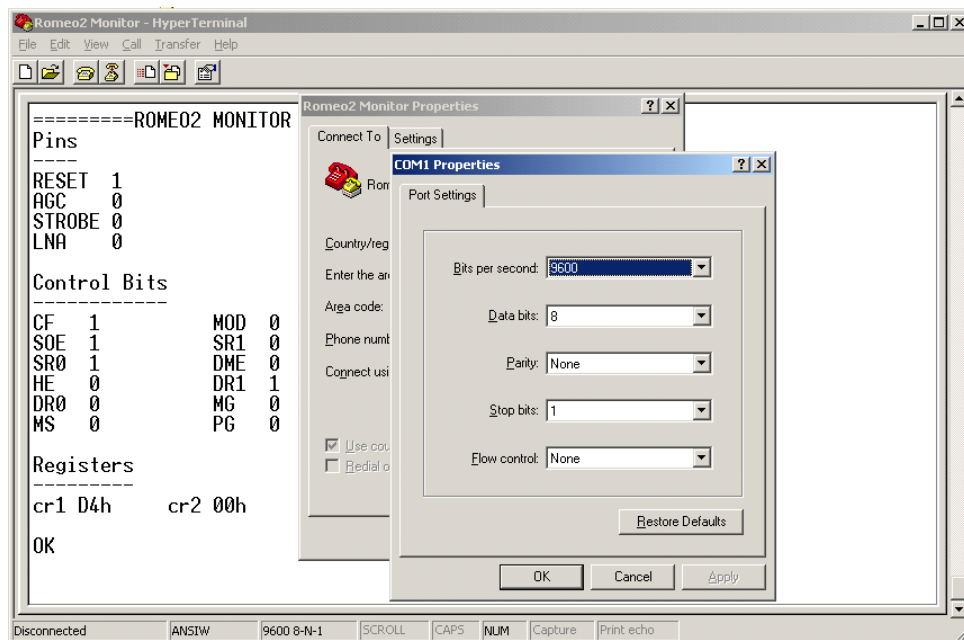


Figure 4. Terminal Program Configuration

To start the monitor:

1. Configure the terminal program for 9600 baud, eight data bits, no parity, one stop bit, no flow control. (If using Hyperterminal, use the Romeo2Monitor.ht setup file.)
2. Connect MC68HC908AP64 Demo Board to the PC comm. port using the 9-pin D connector.
3. Start the terminal program (in Hyperterminal, click the 'Connect' icon on the toolbar at the top of the screen).
4. Connect power to the MC68HC908AP64 Demo Board.

You should now see the screen shown in [Figure 1](#). If not, press the reset switch on the MC68HC908AP64 Demo Board.

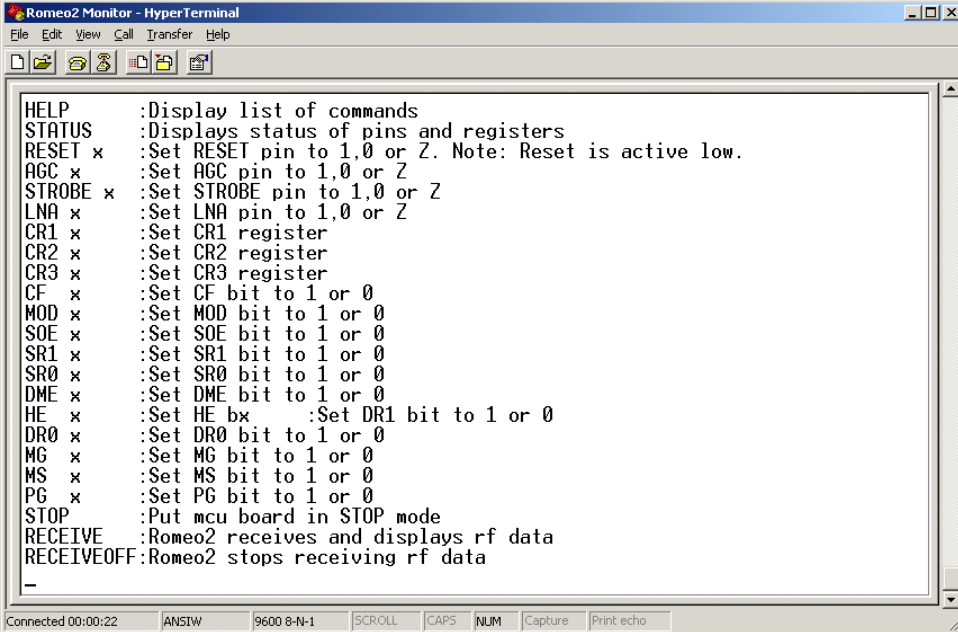
This screen shows the state of various pins that connect between the Romeo2 RF module and the MC68HC908AP64 Demo Board. It also displays the contents of Romeo2's internal registers and displays the state of each bit in these registers.

NOTE

Each Romeo2 RF module is hard-wired to a specific carrier frequency — 315 MHz, 434 MHz, 838 MHz, 915 MHz. Ensure that you program the CF bit in register CR1 to the correct value for your RF module.

Commands can be typed in the terminal program window. Type HELP <return> for a complete list of commands (see [Figure 5](#)).

Command List



```
HELP :Display list of commands
STATUS :Displays status of pins and registers
RESET x :Set RESET pin to 1,0 or Z. Note: Reset is active low.
AGC x :Set AGC pin to 1,0 or Z
STROBE x :Set STROBE pin to 1,0 or Z
LNA x :Set LNA pin to 1,0 or Z
CR1 x :Set CR1 register
CR2 x :Set CR2 register
CR3 x :Set CR3 register
CF x :Set CF bit to 1 or 0
MOD x :Set MOD bit to 1 or 0
SOE x :Set SOE bit to 1 or 0
SR1 x :Set SR1 bit to 1 or 0
SR0 x :Set SR0 bit to 1 or 0
DME x :Set DME bit to 1 or 0
HE x :Set HE bx :Set DR1 bit to 1 or 0
DR0 x :Set DR0 bit to 1 or 0
MG x :Set MG bit to 1 or 0
MS x :Set MS bit to 1 or 0
PG x :Set PG bit to 1 or 0
STOP :Put mcu board in STOP mode
RECEIVE :Romeo2 receives and displays rf data
RECEIVEOFF:Romeo2 stops receiving rf data
-
```

Figure 5. Command List

Command List

The Romeo2 monitor supports the commands listed and explained below.

HELP

Description: Help displays a list of all available commands with a short description of each command.

STATUS

Description: Displays current values of Romeo2 RF module pins and internal register values.

RESET

Description: RESET allows the user to configure the reset pin connection to ROME02. The pin can be set to an output at logic 1 or logic 0, or set to an input (high impedance state).

RESET 1 <return> — Set reset pin to logic 1

RESET 0 <return> — Set reset pin to logic 0

RESET Z <return> — Set pin to input (high impedance state)

AGC

Description: AGC allows the user to configure the Automatic Gain Control (AGC) pin connection to ROMEO2. The pin can be set to an output at logic 1 or logic 0, or set to an input (high impedance state).

AGC 1 <return> — Set AGC pin to logic 1

AGC 0 <return> — Set AGC pin to logic 0

AGC Z <return> — Set pin to input (high impedance state)

STROBE

Description: STROBE allows the user to configure the STROBE oscillator pin connection to ROMEO2. The pin can be set to an output at logic 1 or logic 0, or set to an input (high impedance state).

STROBE 1 <return> — Set STROBE pin to logic 1

STROBE 0 <return> — Set STROBE pin to logic 0

STROBE Z <return> — Set pin to input (high impedance state)

LNA

Description: LNA allows the user to configure the Low Noise Amplifier LNA pin connection to ROMEO2. The pin can be set to an output at logic 1 or logic 0, or set to an input (high impedance state).

LNA 1 <return> — Set LNA pin to logic 1

LNA 0 <return> — Set LNA pin to logic 0

LNA Z <return> — Set pin to input (high impedance state)

CR1

Description: CR1 allows the user to set the value of the CR1 register on ROMEO2. The register can have any value in the range 0x00–0xff.

CR1 value <return>

CR2

Description: CR2 allows the user to set the value of the CR2 register on ROMEO2. The register can have any value in the range 0x00–0xff. CR2 holds the ID value for Romeo2.

CR2 value <return>

Command List

CR3

Description: CR3 allows the user to set the value of the CR3 register on ROMEO2. The register can have any value in the range 0x00–0xff.

CR3 value <return>

CF

Description: CF allows the user to set the value of the Carrier Frequency (CR) bit in register CR1 on ROMEO2. The bit can be set to logic 1 or logic 0.

CF 1 <return> — Set CF bit to logic 1

CF 0 <return> — Set CF bit to logic 0

MOD

Description: MOD allows the user to set the value of the Modulation type (MOD) bit in register CR1 on ROMEO2. The bit can be set to logic 1 or logic 0

MOD 1 <return> — Set MOD bit to logic 1

MOD 0 <return> — Set MOD bit to logic 0

SOE

Description: SOE allows the user to set the value of the Strobe Oscillator Enable (SOE) bit in register CR1 on ROMEO2. The bit can be set to logic 1 or logic 0.

SOE 1 <return> — Set SOE bit to logic 1

SOE 0 <return> — Set SOE bit to logic 0

SR1

Description: SR1 allows the user to set the value of the Strobe Ratio 1 (SR1) bit in register CR1 on ROMEO2. The bit can be set to logic 1 or logic 0.

SR1 1 <return> — Set SR1 bit to logic 1

SR1 0 <return> — Set SR1 bit to logic 0

SR0

Description: SR0 allows the user to set the value of the Strobe Ratio 0 (SR0) bit in register CR1 on ROMEO2. The bit can be set to logic 1 or logic 0.

SR0 1 <return> — Set SR0 bit to logic 1

SR0 0 <return> — Set SR0 bit to logic 0

DME

Description: DME allows the user to set the value of the Data Manager Enable (DME) bit in register CR1 on ROME02. The bit can be set to logic 1 or logic 0.

DME 1 <return> — Set DME bit to logic 1

DME 0 <return> — Set DME bit to logic 0

HE

Description: HE allows the user to set the value of the Header Enable (HE) bit in register CR1 on ROME02. The bit can be set to logic 1 or logic 0.

HE 1 <return> — Set HE bit to logic 1

HE 0 <return> — Set HE bit to logic 0

DR1

Description: DR1 allows the user to set the value of the Data Rate 1 (DR1) bit in register CR3 on ROME02. The bit can be set to logic 1 or logic 0.

DR1 1 <return> — Set DR1 bit to logic 1

DR1 0 <return> — Set DR1 bit to logic 0

DR0

Description: DR0 allows the user to set the value of the Data Rate 0 (DR0) bit in register CR3 on ROME02. The bit can be set to logic 1 or logic 0.

DR0 1 <return> — Set DR0 bit to logic 1

DR0 0 <return> — Set DR0 bit to logic 0

MG

Description: MG allows the user to set the value of the Mixer Gain (MG) bit in register CR3 on ROME02. The bit can be set to logic 1 or logic 0.

MG 1 <return> — Set MG bit to logic 1

MG 0 <return> — Set MG bit to logic 0

RF Message Formats

MS

Description: MS allows the user to set the value of the Mixout Pin Select bit (MS) in register CR3 on ROMEO2. The bit can be set to logic 1 or logic 0.

MS 1 <return> — Set MS bit to logic 1

MS 0 <return> — Set MS bit to logic 0

PG

Description: PG allows the user to set the value of the Phase Comparator Gain (PG) bit in register CR3 on ROMEO2. The bit can be set to logic 1 or logic 0.

PG 1 <return> — Set PG bit to logic 1

PG 0 <return> — Set PG bit to logic 0

STOP

Description: STOP puts the MC68HC908AP64 MCU into its low power STOP mode. This allows the user to make current measurements of the system. To restart the MCU, press the reset button on the MC68HC908AP64 Demo Board.

RECEIVE

Description: RECEIVE configures the system to receive and display RF messages. The reset pin on the Romeo2 RF module is set to logic 1. If Romeo2 receives RF data using any of the data format listed in the Romeo2 data sheet, these will be passed to the MC68HC908AP64 via Romeo2's SPI interface. The MC68HC908AP64 will pass these messages to the terminal program via its serial port. Examples of the RECEIVE command usage are given in the following sections. Pressing the return key will stop the reception of messages.

RECEIVEOFF

Description: RECEIVEOFF stops Romeo2 send RF messages to the display. The reset pin is configured as an output set to logic 0.

RF Message Formats

The Romeo2 monitor allows the user to receive messages using the data formats defined in the Romeo2 data sheet. There are two basic formats: messages with a header field and messages without a header field. [Figure 6](#) shows some basic messages following these formats.



Figure 6. Basic Message Formats

The various fields in the messages are described below.

Preamble — the Preamble is a fixed format field that allows Romeo to detect the start of a message. The Tango3 monitor can transmit Preambles with the correct format using the 'P' field.

ID — Each Romeo device can be assigned an 8-bit ID number. It will receive messages with only this ID. This allows each Romeo device in an RF network to have a unique ID.

The Romeo2 monitor can construct and send any ID byte using the '1' and '0' fields.

NOTE

Romeo2 can also accept messages with TONE based IDs. A TONE is an ID field consisting entirely of 1's or 0's. Refer to the Romeo2 data sheet for more information.

Header — The header field is a 4-bit fixed format field. It notifies Romeo that message data is next. The field has values '0110' or '1001'.

The Romeo2 monitor can construct and send headers using the '1' and '0' fields.

Data — Data can be any length and consists of Manchester encoded '1's and '0's. The Romeo2 monitor can construct any data pattern required.

EOM — End of Message. This is a fixed format field that indicates the end of a message. The Tango3 monitor can transmit EOMs with the correct format using the 'E' field.

The ID, header and data fields are constructed from Manchester encoded 1's and 0's. A Manchester encoded bit is represented by a sequence of two opposite logic levels. A '0' bit of data is encoded as sequence '01', a '1' bit of data is encoded as sequence '10'. [Figure 7](#) shows the data sequence '11001' using Manchester encoding.

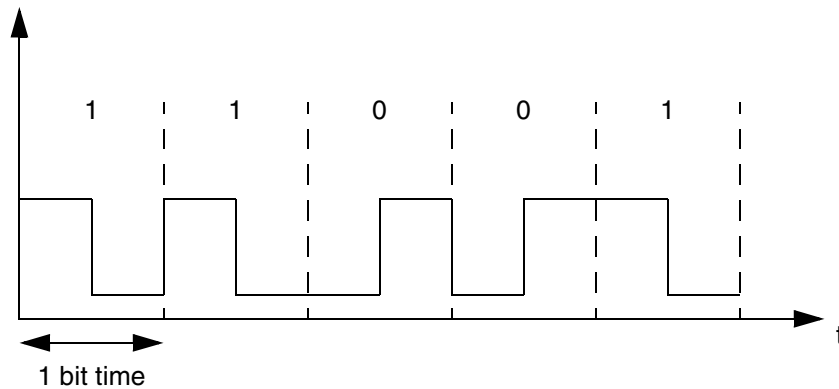


Figure 7. Manchester Encoding Of Data

Examples

Receiving an OOK Message Without Header

Below is an example of receiving an OOK modulated message without a header. The message ID is 0x2a. The data rate is 2400 bits per second, carrier frequency 434 MHz.

To configure system to receive an OOK data frame with ID 0x2a at 434 MHz, no header:

1. Set CR1 register to value 0xDA using command CR1 DA <return>. This configures the following bits:
 CF = 1 (434 MHz carrier frequency)
 MOD = 0 (OOK modulation)
 SOE = 1 (strobe oscillator enabled)
 SR0/1 = 10 (strobe oscillator ratio — value not important)
 DME = 1 (Data Manager Enabled)
 HE = 0 (No header in message)
2. Set CR2 (ID register) to 0x2a using command CR2 2a <return>.
3. Set CR3 register to value 0x48 using command CR3 48 <return>. This configures the following bits:
 DR1/0 = 01 (2–2.7kBd data rate range)
 MG = 0 (normal mixer gain)
 MS = 0 (mixout pin set to mixer output)
 PG = 1 (phase comparator gain = low gain mode)
4. Set LNA pin to logic 1 using command LNA 1 <return>.
5. Set STROBE pin to logic 1 using command STROBE 1 <return>.
6. Set AGC pin to logic 0 using command AGC 0 <return>.
7. Enable reception of messages using command RECEIVE. This sets the reset pin to logic 1, and the Romeo2 monitor will display any received data.

Figure 8 shows the above setup after reception of a message containing one byte of data with value 0xf0 (11110000 binary).

NOTE

When receiving messages where the number of data bits is an exact multiple of eight, Romeo2 may add an extra byte to the end of the message.

```

=====ROME02 MONITOR V1.0 (c)Freescale 2004 =====
Pins
-----
RESET  1
AGC    0
STROBE 1
LNA    1

Control Bits
-----
CF     1      MOD  0
SOE    1      SR1  1
SR0    0      DME  1
HE     0      DR1  0
DR0    1      MG   0
MS     0      PG   1

Registers
-----
cr1 DAh      cr2 2Ah      cr3 48h

Receive Mode Active. Press return to exit
11110000

```

Figure 8. Receiving OOK Message Without Header

Receiving an FSK Message Without Header

Below is an example of receiving an FSK modulated message. The message ID is 0xA5. The data rate is 1000 bits per second, carrier frequency 434 MHz.

1. Set CR1 register to value 0xFA using command CR1 FA <return>
This configures the following bits:
CF = 1 (434 MHz carrier frequency)
MOD = 1 (FSK modulation)
SOE = 1 (strobe oscillator enabled)
SR0/1 = 10 (strobe oscillator ratio — value not important)
DME = 1 (Data Manager Enabled)
HE = 0 (No header in message)
2. Set CR2 (ID register) to 0xA5 using command CR2 a5 <return>.
3. Set CR3 register to value 0x08 using command CR3 08 <return>.
This configures the following bits:
DR1/0 = 00 (1–1.4kBd data rate range)

RF Message Formats

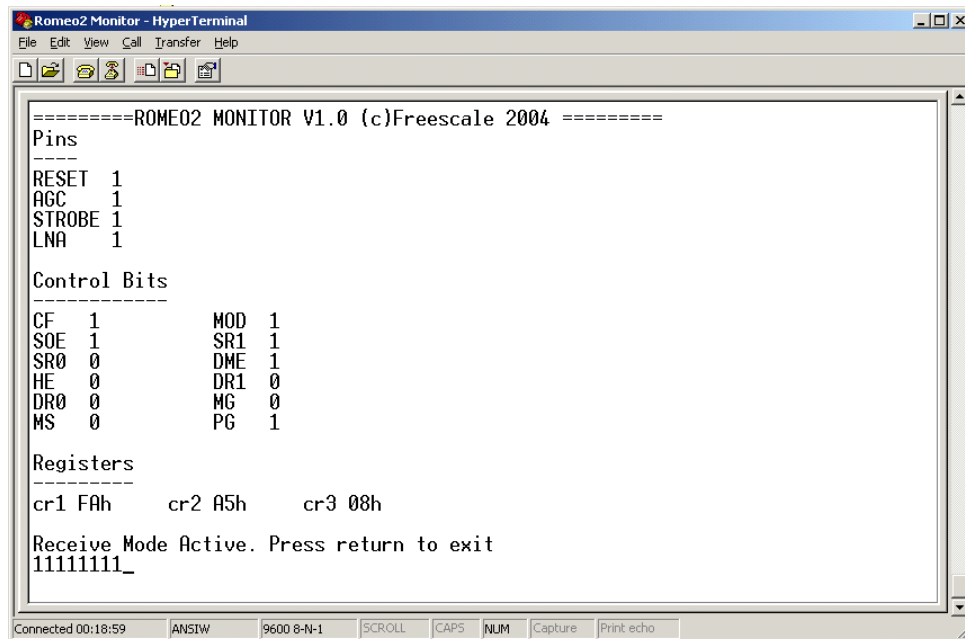
MG = 0 (normal mixer gain)
MS = 0 (mixout pin set to mixer output)
PG = 1 (phase comparator gain = low gain mode)

4. Set LNA pin to logic 1 using command LNA 1 <return>.
5. Set STROBE pin to logic 1 using command STROBE 1 <return>.
6. Set AGC pin to logic 1 using command AGC 1 <return>.
7. Enable reception of messages using command RECEIVE. This sets the reset pin to logic 1, and the Romeo2 monitor will display any received data.

Figure 9 shows the above setup after reception of a message containing one byte of data with value 0xff (11111111 binary).

NOTE

When receiving messages where the number of data bits is an exact multiple of eight, Romeo2 may add an extra byte to the end of the message.



```
=====ROME02 MONITOR V1.0 (c)Freescale 2004 =====
Pins
-----
RESET 1
AGC 1
STROBE 1
LNA 1

Control Bits
-----
CF 1          MOD 1
SOE 1         SR1 1
SR0 0         DME 1
HE 0          DR1 0
DR0 0         MG 0
MS 0          PG 1

Registers
-----
cr1 FAh      cr2 A5h      cr3 08h

Receive Mode Active. Press return to exit
11111111_
```

Figure 9. Receiving FSK Message Without Header

Receiving Messages With Header Field

To receive a message with an ID of 0x55 (01010101 binary), with a header field (0110 binary), at data rate of 1000 bps, OOK modulation, 434 MHz carrier frequency:

1. Set CR1 register to value 0xDB using command CR1 DB <return>. This configures the following bits:

- CF = 1 (434 MHz carrier frequency)
MOD = 0 (OOK modulation)
SOE = 1 (strobe oscillator enabled)
SR0/1 = 10 (strobe oscillator ratio — value not important)
DME = 1 (Data Manager Enabled)
HE = 1 (Header enabled)
2. Set CR2 (ID register) to 0x55 using command CR2 55 <return>.
 3. Set CR3 register to value 0x08 using command CR3 08 <return>.
This configures the following bits:
DR1/0 = 00 (2–2.7kBd data rate range)
MG = 0 (normal mixer gain)
MS = 0 (mixout pin set to mixer output)
PG = 1 (phase comparator gain = low gain mode)
 4. Set LNA pin to logic 1 using command LNA 1 <return>.
 5. Set STROBE pin to logic 1 using command STROBE 1 <return>.
 6. Set AGC pin to logic 0 using command AGC 0 <return>.
 7. Enable reception of messages using command RECEIVE. This sets the reset pin to logic 1, and the Romeo2 monitor will display any received data.

Figure 10 shows the above setup after reception of a message containing one byte of data with value 0xff (11111111 binary).

NOTE

When receiving messages where the number of data bits is an exact multiple of eight, Romeo2 may add an extra byte to the end of the message.

```

=====ROME02 MONITOR V1.0 (c)Freescale 2004 =====
Pins
-----
RESET 1
AGC 0
STROBE 1
LNA 1

Control Bits
-----
CF 1          MOD 0
SOE 1        SR1 1
SR0 0        DME 1
HE 1         DR1 0
DR0 0        MG 0
MS 0         PG 1

Registers
-----
cr1 DBh      cr2 55h      cr3 08h

Receive Mode Active. Press return to exit
11111111

```

Figure 10. Receiving Message With Header Field

Receiving Data Without Manchester Encoding

The Romeo2 monitor can be used to configure Romeo2 to receive RF data not encoded using Manchester encoding. To do this, Romeo2's Data Manager should be disabled (set DME bit to logic 0). However, this data will not be displayed correctly on the terminal screen. The user should use an oscilloscope on Romeo2's MOSI pin to monitor the data.

For example, to receive data in different encoding formats at 434 MHz, OOK modulation:

1. Set CR1 register to value 0xD8 using command CR1 DA <return>. This configures the following bits:
 CF = 1 (434 MHz carrier frequency)
 MOD = 0 (OOK modulation)
 SOE = 1 (strobe oscillator enabled)
 SR0/1 = 10 (strobe oscillator ratio — value not important)
 DME = 0 (Data Manager Enabled)
 HE = 0 (No header in message)
2. Set CR3 register to value 0x48 using command CR3 48 <return>. This configures the following bits:
 DR1/0 = 01 (2–2.7kBd data rate range)
 MG = 0 (normal mixer gain)
 MS = 0 (mixout pin set to mixer output)
 PG = 1 (phase comparator gain = low gain mode)
3. Set LNA pin to logic 1 using command LNA 1 <return>.
4. Set STROBE pin to logic 1 using command STROBE 1 <return>.

- Set reset pin to logic 1 using command RESET 1 <return>.

```

=====ROME02 MONITOR V1.0 (c)Freescale 2004 =====
Pins
-----
RESET  1
AGC    0
STROBE 1
LNA    1

Control Bits
-----
CF     1          MOD  0
SOE    1          SR1  1
SR0    0          DME  0
HE     0          DR1  0
DR0    0          MG   0
MS     0          PG   1

Registers
-----
cr1 D8h   cr2 00h   cr3 08h

OK
  
```

Figure 11. Receiving Data Without Manchester Encoding

Romeo2 Monitor Source Code

The Romeo2 monitor was written using the C programming language. Full source code is supplied in file `Romeo2AP64MonSource.zip`, which can be downloaded from www.freescale.com.

Modifying Romeo2 Monitor

The Romeo monitor program has a simple structure. It has been written to allow easy porting to other MCUs. The code is liberally commented and should be easy to understand.

Decoding of commands typed into the PC terminal program is done in the `SCIRx()` routine. This routine is called each time the MCU receives a character from the PC keyboard. It stores characters in a buffer until a carriage return is detected. It then decodes the buffer and performs the required function for each message.

When configuring the internal registers on Romeo2, the monitor uses the MCU's SPI peripheral. It also uses the SPI when the `RECEIVE` command is used to receive data from Romeo2.

The monitor uses the following MCU resources.

- SPI 1
- SCI 1
- FLASH
- RAM

It should be possible to port the code to another MCU by changing the register definitions and some other definitions at the beginning of the code.

References

1. MC33493 RF Transmitter IC (Tango3) data sheet
2. MC33591 RF Receiver (Romeo2) data sheet

Trademarks

- Windows[®] is a registered trademark of Microsoft Corporation in the U.S. and/or other countries.
- CodeWarrior[®] is a registered trademark of MetroWerks, Inc., a wholly owned subsidiary of Motorola, Inc.
- MetroWerks[®] and the MetroWerks logo are registered trademarks of MetroWerks, Inc., a wholly owned subsidiary of Motorola, Inc.

How to Reach Us:

USA/Europe/Locations not listed:
Freescale Semiconductor Literature Distribution
P.O. Box 5405, Denver, Colorado 80217
1-800-521-6274 or 480-768-2130

Japan:
Freescale Semiconductor Japan Ltd.
SPS, Technical Information Center
3-20-1, Minami-Azabu
Minato-ku
Tokyo 106-8573, Japan
81-3-3440-3569

Asia/Pacific:
Freescale Semiconductor H.K. Ltd.
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T. Hong Kong
852-26668334

Learn More:
For more information about Freescale Semiconductor products, please visit
<http://www.freescale.com>

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale[™] and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.
© Freescale Semiconductor, Inc. 2004.