Order Number: AN2185/D
Rev. 0, 08/2001

AN2185

Application Note

MCU Interface Using Romeo2

Using the SPI

By using a Serial Peripheral Interface (SPI) interrupt service routine to fetch UHF data, a greatly reduced cpu load is realized. The SPI also allows the UHF data reading to run as a background task.

The following is an example of what Romeo2 output might look like while reading software with an HC908.

```
Init_Read_UHF      Idhx #first_byte

Read_Tx_UHF        wait
                   cbeqx #last_byte,end_Tx_read
                   jmp Read_Tx_UHF

SPI_ISR            Ida spsr
                   MOV SPDR,X+
                   rti
```

with “last_byte” being the address of the last UHF byte,
“first_byte” being the address of the first UHF byte,
“spsr” being the SPI status and control register,
“spdr” being the SPI register.

Therefore, to read a byte from Romeo2, the cpu is busy for only 24 cpu cycles, or 3 µs at 8 MHz. This means that the cpu is busy for only 0.3% of the time with a 9600 bauds datarate.

Standby Mode

There are two options which can occur while in standby mode.

1. The MCU is stopped.
2. The MCU is in Wait mode at a low cpu speed.

Option 1

If an HC908 MCU is in Stop mode, it can only be awaken by a falling edge on the IRQ or reset pins.

Let us make the following assumptions:

- Datarate = 9600 bauds
- Romeo2 strobe period = 4 ms

One solution is to route the SCLK of Romeo2's output to the IRQ and SCLK pins of the MCU.

The protocol is:

- Wake-up signal (tone)
- Header
- Useful Data

Romeo2 then processes the received UHF data in two steps. First, Romeo2 will initially be set up as DME = 1 and HE = 0. The tone length must be longer than the total of:

Two successive Romeo2 wake-up periods + the MCU wake-up time + one byte length

As soon as the received UHF frame awakens Romeo2, a clock signal is shifted out of Romeo2. The SCLK output from Romeo2 is routed to the SCLK of the MCU and also to the IRQ. The first edge on the SCLK then wakes up the MCU from Stop mode. Once the MCU is running, the IRQ is disabled so that the IRQ interrupt requests are no longer serviced during data reception and the MCU will read data on MOSI, thanks to the SPI.

The tone is long enough so that at least one $FF byte is read by the SPI MCU. A time-out, thanks to the PIT, can also be run in parallel so that the MCU goes back to sleep if it was a false wake-up signal (for example, a glitch on the SCLK). If a $FF is effectively read on MOSI, it means that the MCU received a valid UHF wake-up signal.

Romeo2 now goes on to the second step. In this step, the MCU changes the Romeo2 set-up so that HE = 1 and holds Romeo2 awake with Strobe = High. Useful data is now read with its SPI port. Another MCU time-out might be run so that the MCU goes back into Sleep mode if no data is received within a certain time-slot. When the time-out occurs or when the data is fully received, the initial Romeo2 set-up is downloaded back.

Both the above protocol and this type of data handling was demonstrated experimentally by Freescale.

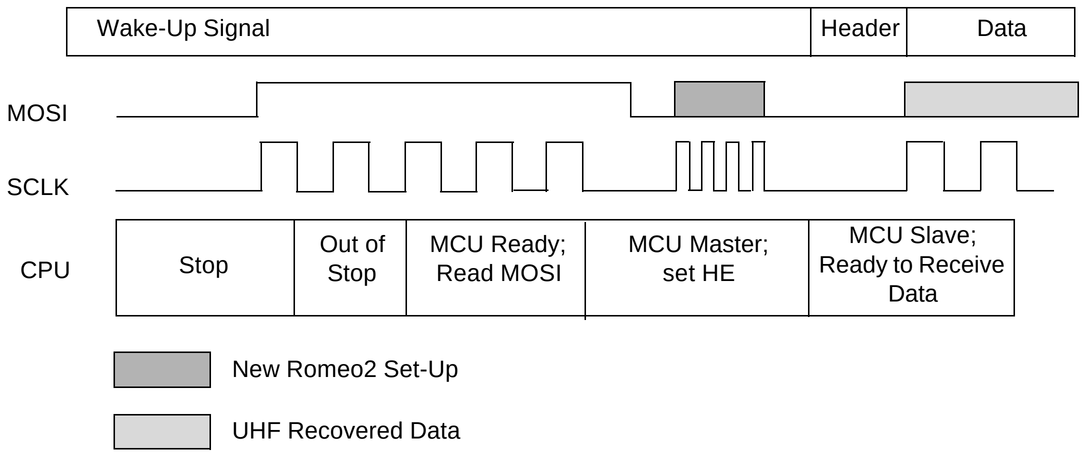

Figure 1.

Note that Figure 1 is true when there is no pull-up on IRQ. If there is a pull-up on IRQ, the wake-up sequence is a little different. As a matter of fact, Romeo2 pulls down SCLK and MOSI when it is awaken by its internal strobe oscillator, causing a falling edge on SCLK. The result is that the MCU is awaken at each Romeo2's wake-up. The MCU is now in Wait mode for the time-out duration. If no data is received within this time slot, the MCU goes back to sleep.

To evaluate the total I_{dd} current in parking mode, assume the following:

With no pull-up on IRQ,

I_{dd}_HC08 = 100 µA in stop (worst case stop current of an HC08AB32 MCU.)

I_{dd}_Romeo2 = 0.45 mA with an on/off ratio of 15

Total I_{dd} = 0.55 mA

With a pull-up on IRQ, the MCU is awaken and in wait for the time-out length. Although the length can be as short as one byte, in this case a 2 ms time-out will be considered so that 1 ms is taken into account for servicing the pending software jobs. It will also be assumed that a 4 MHz crystal is used so that the maximum cpu frequency can be 16 MHz. When exiting out of stop, the cpu frequency is F_crystal/4 = 1 MHz; in this evaluation, 1ms is assumed as the time to exit out of stop. The wait I_{dd} at 1 MHz is 0.8 mA (2H56A mask set).

Mean I_{dd}_HC08 = 0.8mA*3/64 + 0.1 mA*61/64 = 0.13 mA

(the MCU is in Wait for 3 ms which is split into 2 ms for the time-out and 1 ms for the MCU wake-up time, and in Stop for 61 ms; 64 ms being the total Romeo2 wake-up period.)

I_{dd}_Romeo2 = 0.45mA with an off/on ratio of 15

Total I_{dd} = 0.6 mA

Option 2

With this option, the MCU goes into Wait mode with the SPI enabled in order to wake up the MCU. Freescale has tested this option using an HC08 (mask set 2H56A) and a Romeo2 RF receiver. Although the MCU is in wait mode, the MCU is configured at the lowest cpu speed that is compatible for reading data. In fact, it is assumed that when a vehicle is parked, only the receiver function is required for the body controller and that no high cpu speed is needed.

The time necessary for the MCU to read and save the received byte is 22 cpu cycles. This time slot has to be smaller than the byte time slot. This means that 22*T_cpu < T_Byte. Therefore, T_cpu must be smaller than 37 µs. The lowest cpu frequency which is required for reading Romeo2 output, utilizing the SPI, is then about 30 kHz.

Assuming that the MCU quartz is 1 MHz so that it can speed up to 4 MHz with the internal PLL, when the PLL is disabled, the cpu frequency is then 250 kHz. This is higher than the minimum cpu frequency which is required to read data at 9600 bauds. This scenario has also been demonstrated experimentally.

I_{dd}_HC08 = 0.56 mA in wait with a 1 MHz quartz (and a 250 kHz cpu clock)
and the SPI module enabled (mask set 2H56A)
I_{dd}_Romeo2 = 0.45 mA with an on/off ratio of 15

Total I_{dd} in parking with the HC08 = 1.01 mA

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor