NXP

Freescale Semiconductor

by S. Robb
   8/16-bit MCU Division
   Freescale, East Kilbride

Introduction

The Freescale Scalable Controller Area Network (MSCAN) is the specific implementation of the Freescale Scalable CAN concept. The MSCAN module is integrated onto several families of Freescale microcontrollers, including the MC68HC08, MC68HC12 and MC9S12 families. The MSCAN module fully implements the CAN 2.0 A/B protocol as defined in the Bosch specification of September 1991. For readers to fully understand the MSCAN behaviour, it is recommended that the Bosch specification and the MSCAN user guide are consulted. The MSCAN specification varies slightly for different microcontrollers, so the user guide or technical data appropriate to the microcontroller of interest should be consulted.

The MSCAN module on the microcontroller is connected to the CAN bus by means of a physical interface. The CAN bus itself is usually a differential twisted pair of wires, although single wire implementations also exist. For ease of use, several CAN interface integrated circuits are available. The CAN interfaces can be divided into two basic types: high speed and fault tolerant. High speed CAN interfaces can operate over the full range of CAN bus speeds, from 10kbps (10,000 bits per second) to 1Mbps (1000,000 bits per second). In contrast, fault tolerant CAN interfaces are limited to 125kbps. Fault tolerant CAN interfaces continuously monitor the CAN bus lines for faults, such as a broken wire or a wire shorted to battery voltage, ground, or both wires shorted together. If a fault is detected, the fault tolerant CAN interface will automatically take action to minimise the system current consumption and if possible, to allow communication to continue on a single wire. In case of a fault recovery, the fault tolerant CAN interface will automatically switch back to normal operation. An example of a fault tolerant CAN interface is the MC33388. A high speed CAN interface offers no fault tolerant features.

freescale
semiconductor

**For More Information On This Product,
Go to: www.freescale.com**

Freescale Semiconductor, Inc.

A higher level of integration is available with 'system basis' chips. System basis chips integrate many circuits commonly found in embedded CAN applications, offering a saving in circuit board area and cost. A system basis chip such as the MC33889 integrates many functions including a fault tolerant CAN interface, dual voltage regulator, supply voltage monitor, programmable watchdog and wake-up features. The system basis chip is controlled by a SPI interface to the microcontroller. The MC33989 is similar but with a high speed CAN interface.

The microcontroller, MSCAN module, CAN interface and system basis chips all have low power modes. The lowest power consumption mode is usually called 'sleep' mode. The purpose of low power modes is to reduce power consumption to a minimum during periods of inactivity. Minimising power consumption is important for battery powered applications. For example, an automotive application is battery powered whilst the engine is switched off and there must be sufficient charge left in the battery to start the engine after a period of at least two weeks. During periods of inactivity, an electronic control unit (ECU) will be predominantly in a low power sleep mode. The ECU may be required to wake-up periodically to check for new activity or in response to an external stimulus either on an input pin or on the CAN bus. The purpose of this paper is to describe the co-ordinated actions necessary to put a CAN application into a low power sleep mode such that it will wake-up in response to a specified stimulus. The description in this paper is based on the MC9S12D family of microcontrollers, however it also applies to the MC68HC08 and MC68HC12 microcontrollers that have an MSCAN module, albeit with slightly different register and register bit names.

Low Power Modes

Microcontroller   A microcontroller has three basic power consumption modes: normal, wait and stop.

In normal mode the microcontroller is fully operational. This mode has the highest power consumption. Certain unused peripheral modules may be held in an initialisation mode, preventing unnecessary power consumption.

In wait mode, clocks to the CPU are stopped, substantially reducing power consumption. Clocks to peripheral modules may also be stopped under software control, further reducing power consumption. Wait mode is entered by executing the WAI instruction. Wait mode is exited when an unmasked interrupt is detected, either from an external signal or from one of the peripheral modules that is still running, or by a reset.

In stop mode, all internal clocks are stopped. The external crystal oscillator is also stopped (unless the PSTP bit in the CLKSEL register is set). Power consumption is reduced to an absolute minimum. Stop mode is entered by executing the STOP instruction with the S bit in the condition codes register (CCR) clear. Stop mode is exited when an unmasked interrupt due to an external signal is detected, or by a reset. A message on the CAN bus is capable of generating a wake-up interrupt, if configured correctly. On exit from stop mode the crystal oscillator is restarted. If the PSTP bit in the CLKSEL register is set when the STOP instruction is executed the oscillator is not stopped. This results in greater power consumption but with the advantage that the crystal start up delay is eliminated and also reduces the mechanical stress and ageing of the crystal in case of frequent STOP conditions.

One other factor that has a significant effect on power consumption is the microcontroller bus frequency. The microcontroller typically uses an external crystal oscillator with a frequency in the range of 1MHz to 8MHz. An internal phase locked loop (PLL) then generates the internal bus frequency of up to 25MHz. An effective way to reduce power consumption is to disable the PLL which will reduce the internal bus frequency to half the crystal oscillator frequency. This technique is commonly used in conjunction with wait mode. Peripheral modules that are clocked from the same source as the internal bus clock will have their timing affected. The MSCAN module can be clocked directly from the crystal oscillator and will not be affected if this option is selected.

MSCAN

The MSCAN module has two low power modes: Sleep mode and Power-down mode. In Sleep mode power consumption is reduced by stopping all MSCAN clocks except those required for the CPU to access registers. In Power Down mode, all MSCAN clocks are stopped; this is the lowest power consumption mode. If the MSCAN is disabled (CANE = 0) all MSCAN clocks except those required for the CPU to access registers are stopped to reduce power consumption. **Table 1 . CPU versus MSCAN Operating Modes** summarises the combinations of MSCAN and CPU modes.

Table 1. CPU versus MSCAN Operating Modes

CPU Mode	MSCAN Mode		
	Normal	**Sleep**	**Power Down**
RUN	CANE = 1 CSWAI = X SLPRQ = 0 SLPAK = 0	CANE = 1 CSWAI = X SLPRQ = 1 SLPAK = 1	
WAIT	CANE = 1 CSWAI = 0 SLPRQ = 0 SLPAK = 0	CANE = 1 CSWAI = 0 SLPRQ = 1 SLPAK = 1	CANE = 1 CSWAI = 1
STOP			CANE = X CSWAI = X SLPRQ = X SLPAK = X

Note: 'X' means 'don't care'.

As can be seen from **Table 1 . CPU versus MSCAN Operating Modes**, the MSCAN can be in normal mode or Sleep mode whilst the CPU is running. The MSCAN registers are configured whilst the CPU is running to select the required mode when the CPU enters Wait mode. Power Down mode is always entered when the microcontroller enters Stop mode.

An MSCAN Wake-up interrupt can only occur if:

- The MSCAN is in Sleep mode or Power Down mode with SLPRQ = 1 and SLPAK = 1,
- Wake-up functionality is enabled (WUPE = 1),
- The Wake-up interrupt is enabled (WUPIE = 1),
- Interrupts are not masked (I = 0 in CCR) and
- A dominant level is detected at the RX input (required duration dependant on WUPM bit).

The MSCAN will not receive or acknowledge the first CAN message which generates the Wake-up interrupt. Furthermore, if the microcontroller was in Stop mode when the CAN message was received, no CAN messages will be received or acknowledged until the microcontroller oscillator has restarted and the MSCAN has synchronised to the CAN bus (synchronisation requires that 11 consecutive recessive bits are detected). In a network in which all nodes are in a low power mode, the first message to be transmitted will serve to wake-up all nodes that are appropriately configured, but the message may only be received and acknowledged by the node that recovers to normal operation the quickest. Thus a second message may need to be transmitted if confirmation that all nodes are operational is required.

CAN Interface

The CAN interface or system basis chip will also have a low power mode, usually called Sleep or Standby mode. The operational mode of a CAN interface is determined by the voltage on one or two pins. These pins are usually connected to I/O pins of the microcontroller so that the microcontroller can control the operational mode of the CAN interface, as shown in **Figure 1 . Typical CAN Application Circuit**. The operational mode of a system basis chip is determined by the value of an internal register; the microcontroller controls the operational mode by sending appropriate commands via the SPI interface.

The microcontroller thus controls the operational mode of the CAN interface or system basis chip, placing the CAN interface in a low-power sleep mode prior to a period of CAN inactivity and returning the CAN interface to normal operation when CAN communication is required. The trigger to wake-up may be the receipt of a CAN message, and the CAN interface or system basis chip is able to detect the CAN message whilst in sleep mode. Dominant bits on the CAN bus will cause the CAN interface or system basis chip to drive a low level (dominant value) on its RX output, so enabling the appropriately configured microcontroller to wake-up. System basis chips and some CAN interfaces also have a separate output pin that can be used to wake-up the microcontroller via an interrupt input pin. The system basis chip can use this output to wake-up the microcontroller on a periodic basis if desired.
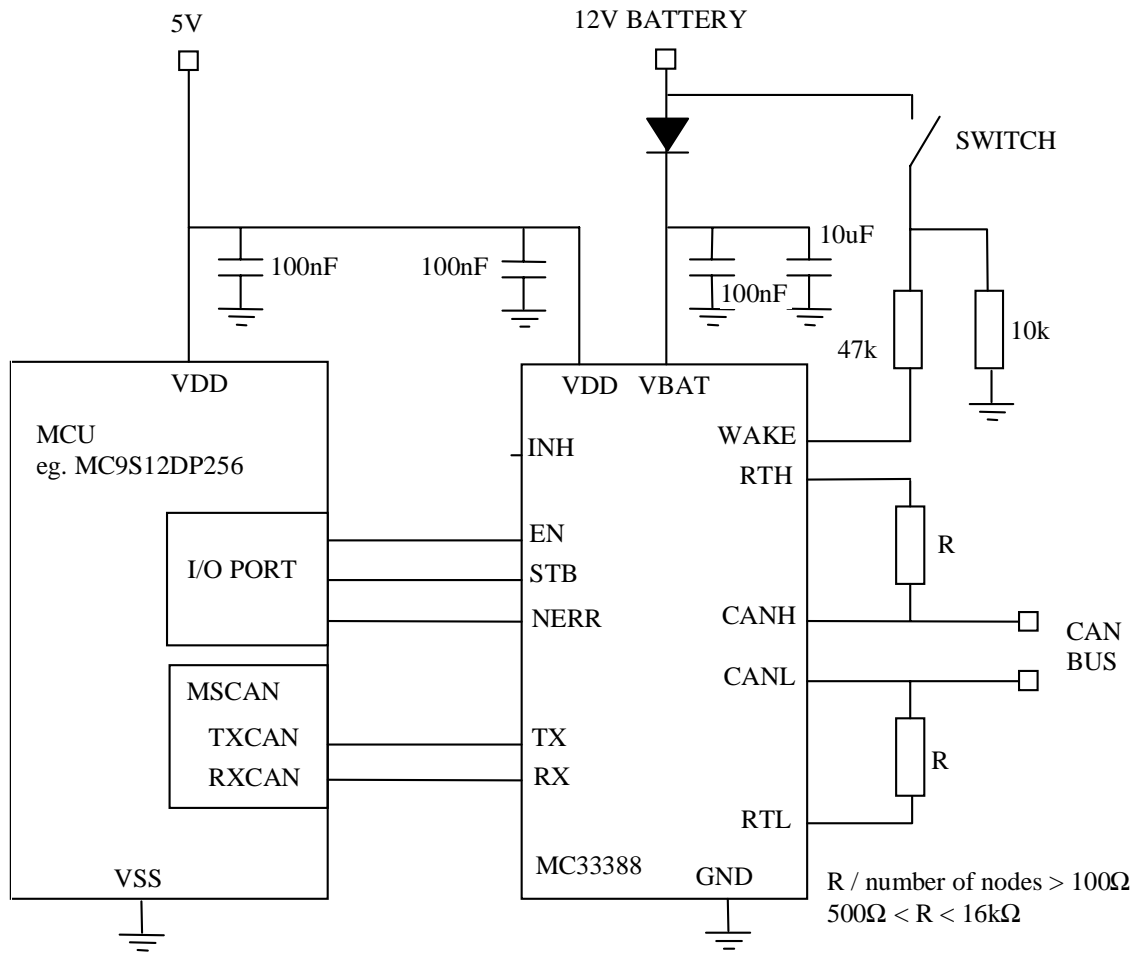
Figure 1. Typical CAN Application Circuit

Entering Low Power Mode

NOTE: *In the following description, 'clear' means bit = 0, and 'set' means bit = 1.*

In order to ensure reliable wake-up operation and to prevent generation of CAN errors, an appropriate procedure must be followed when entering low power mode.

1. First it is necessary to put the MSCAN into sleep mode and configure the wake-up options while the CPU is still running. To enter sleep mode, the SLPRQ bit in the CANCTL0 register is set by the CPU. Any pending message transmissions should either be transmitted in their entirety or should be aborted before the SLPRQ bit is set by the CPU.

NOTE: *Sleep mode is not entered immediately!*

If the CAN bus is quiet, sleep mode will be entered within 1 or 2 bit times. If there is message being transmitted or received, sleep mode will be entered when the message transmission has completed.

When the MSCAN has entered sleep mode, the SLPAK bit in the CANCTL1 register is set by the MSCAN. The MCU must wait for the SLPAK to become set before proceeding to the next step.

If another message follows, the MSCAN will immediately wake-up! Thus, it is suggested that the CAN network operational modes are handled by network management software. For example, a specific message can be broadcast telling every node to go into sleep mode at the same time.

2. The CAN physical interface or system basis chip must now be commanded to enter its low power mode. The physical interface must not be put in a low power mode before the MSCAN is in sleep mode. If the periodic wake-up feature of the system basis chip is to be enabled, this must be done before the sleep command is sent.

3. MCU wake-up functionality must now be enabled. If the MSCAN will be used to wake-up the MCU, refer to 3a. If the MCU will be awakened with an interrupt pin connected to the CAN interface or system basis chip, refer to 3b.

3a. Using the MSCAN to wake-up the MCU.
The WUPE bit in the CANCTL0 register must be set, to enable wake-up on bus activity. The WUPIE bit in the CANRIER register must be set and there must be an interrupt routine to handle the wake-up interrupt. The I-mask bit in the CCR must be cleared. The WUPM bit in the CANCTL1 register may be set to filter out any short spikes from the CAN interface RX pin.

3b. Using an Interrupt pin to wake-up the MCU
The appropriate pin from the CAN interface or system basis chip (e.g. NERR or INT) must be connected to a port pin with interrupt functionality e.g. PORTP.
The port pin must be configured as an input, the interrupt enable bit for the port pin must be set, and the port pin must be configured to generate an interrupt on a falling edge. There must be an interrupt routine to handle the port interrupt. The I-mask bit in the CCR must be cleared.

4. If SLPAK is still set, the MCU can now be put in a low power mode, either Wait or Stop.
If the MCU enters Stop mode, the MSCAN enters 'power-down' mode. If the MCU enters Wait mode, the MSCAN enters 'power-down' mode if the CSWAI bit in the CANCTL0 register was previously set, or remains in sleep mode if the CSWAI bit is clear.
The S bit in the Condition Codes Register (CCR) must be clear for the STOP instruction to be executed.
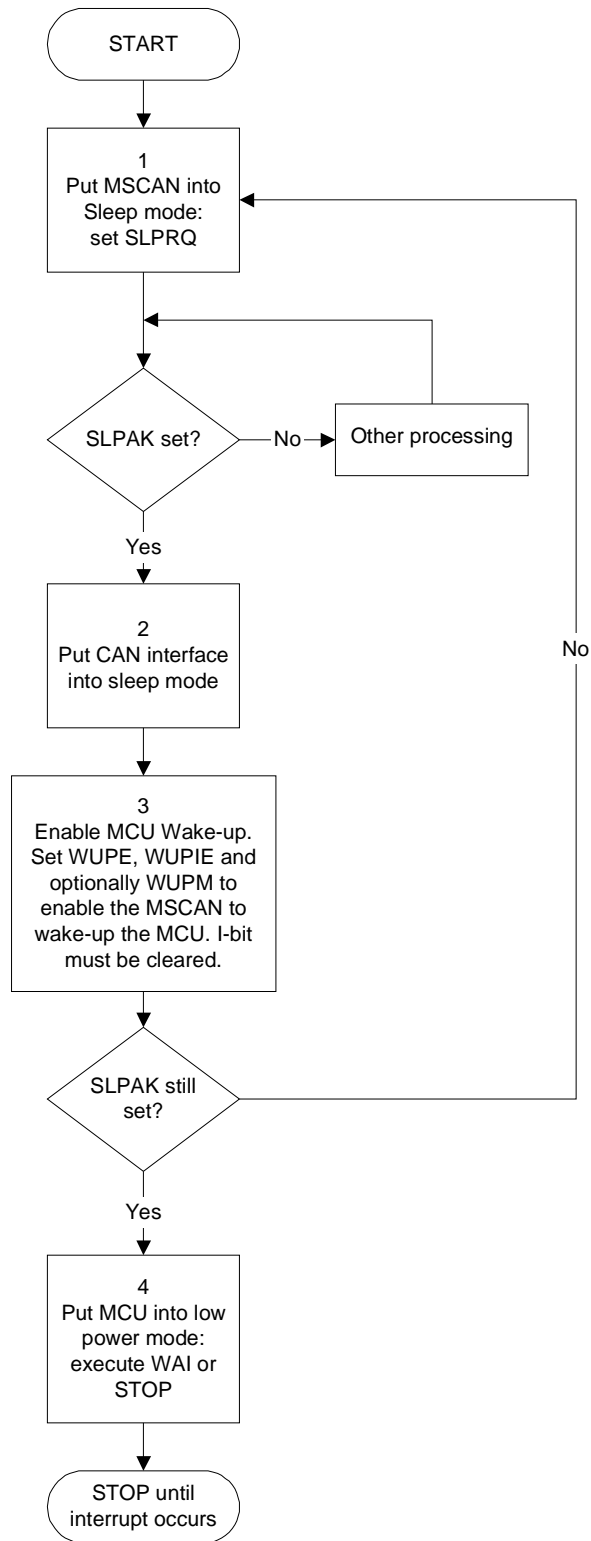
Freescale Semiconductor, Inc.

```
                            ┌──────────┐
                            │  START   │
                            └────┬─────┘
                                 │
                    ┌────────────▼────────────┐
                    │            1            │◄──────────────┐
                    │   Put MSCAN into        │               │
                    │   Sleep mode:           │               │
                    │   set SLPRQ             │               │
                    └────────────┬────────────┘               │
                                 │                            │
                                 ▼◄──────────────┐            │
                              ◇                  │            │
                        SLPAK set? ──No──► Other processing   │
                              ◇                                │
                                 │Yes                          │
                    ┌────────────▼────────────┐                │
                    │            2            │                │
                    │   Put CAN interface     │                │
                    │   into sleep mode       │                │
                    └────────────┬────────────┘                │
                                 │                             │
                    ┌────────────▼────────────┐                │
                    │            3            │                │
                    │   Enable MCU Wake-up.   │                │
                    │   Set WUPE, WUPIE and   │                │
                    │   optionally WUPM to    │                │
                    │   enable the MSCAN to   │                │
                    │   wake-up the MCU. I-bit│                │
                    │   must be cleared.      │                │
                    └────────────┬────────────┘                │
                                 │                             │
                              ◇                                │
                         SLPAK still ──────No─────────────────┘
                            set?
                              ◇
                                 │Yes
                    ┌────────────▼────────────┐
                    │            4            │
                    │   Put MCU into low      │
                    │   power mode:           │
                    │   execute WAI or        │
                    │   STOP                  │
                    └────────────┬────────────┘
                                 │
                            ┌────▼─────┐
                            │ STOP until│
                            │ interrupt │
                            │  occurs   │
                            └──────────┘
```

Figure 2. Enter Low Power Mode Flowchart

MSCAN Wake-up

When a message is transmitted on the CAN bus by another node, the sleeping MCU will be awakened by the interrupt selected in step 3. If the MCU is awakened by the MSCAN then the MSCAN will have automatically exited from its sleep mode.

If the MCU is awakened by another source, such as an interrupt pin, the MSCAN will remain in its sleep mode until the CPU clears the SLPRQ bit in the CANCTL0 register and the MSCAN clears the SLPAK bit in the CANCTL1 register in response. In both cases the MCU must bring the CAN interface or system basis chip out of its low power mode in order to receive or transmit CAN messages. When the code in the appropriate interrupt service routine has executed, code execution continues from the last instruction before the interrupt occurred. If the MCU was in Stop or Wait mode, code execution continues with the next instruction after the WAI or STOP instruction.

The first message transmission serves to wake-up the sleeping MCU, but is not received. This message will be automatically retransmitted until it is acknowledged by one node on the network. Note that some nodes may take longer to wake-up than others. In particular, MCUs that are in STOP mode will have to restart the oscillator before messages can be received (if the PSTP bit in the CLKSEL register is set, the oscillator continues to run in STOP mode which reduces the start-up time - this is called pseudo-STOP mode).

```
        ( MSCAN Wake-up
          Interrupt )
               |
               v
      +--------------------+
      | Clear WUPIF, WUPIE |
      |   Clear SLPRQ      |
      +--------------------+
               |
               v
      +--------------------+
      |    Enable CAN      |
      |    Interface       |
      +--------------------+
               |
               v
        ( Return from
          Interrupt. )
```
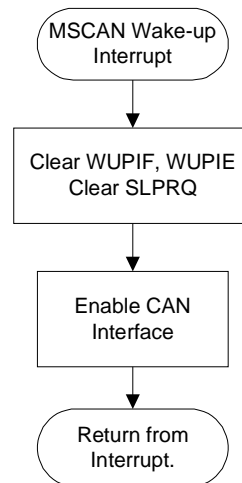
Figure 3. MSCAN Wake-up Interrupt Flowchart

This Page Has Been Intentionally Left Blank

Freescale Semiconductor, Inc.

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