Application Note

AN2461/D
Rev. 0, 2/2003

Low Power Management using HCS12 and SBC devices

By Manuel Alves
Field Application Engineer
Transportation and Standard Products Group
Toulouse, France

Introduction

Low power consumption is an important consideration in a wide range of applications. This is even a critical parameter in automotive applications in order to prevent high battery discharge rates and conserve the battery charge between vehicle starts (recharge cycles).

Car manufacturers expect that a car can still start after six weeks in “sleep mode”. Therefore each Electronic Control Unit, depending on its functionality, has its own low-power requirements. As a consequence of the increase of electronic contents into the car, quiescent current specifications become harder and harder to achieve. Automotive OEMs (original equipment manufacturers) require power consumption profiles that can hardly be met using “standard”, non-integrated devices.

This document describes different low-power strategies and how they can be implemented easily using an HCS12 microcontroller together with a System Basis Chip (SBC) device from Freescale. SBC is a generic term for devices that integrate several standard functionalities such as voltage regulators with reset controller, bus physical interface, high voltage inputs, high-side switch, flexible watchdog etc. An SBC is configured via a Serial Peripheral Interface (SPI) and reports diagnostic to the microcontroller via the same link.

Low power modes

For any electronic device, the current consumption in low power modes is always a compromise with the wake-up capabilities. The fewer wake-up capabilities, the lower the quiescent current. For the large majority of automotive applications the wake-up conditions are:

- Bus activity
- External event (switched contact to battery voltage or GND)

Some specific applications may require one of the following additional features:

- Cyclic auto wake-up in order to read analog inputs and/or refresh a watchdog
- Very fast wake-up timings in order to interpret serial bus communication or to operate an actuator very quickly (door unlocking for instance)

The sections hereafter detail the low power modes against wake-up capabilities for the HCS12 and the SBC individually. This section is intended to highlight the key differences between low power modes for each device. Please refer to individual product specifications for more detailed information.

HCS12 Microcontroller

Like on any CMOS integrated circuit, the power consumption of a microcontroller depends on its operation frequency, knowing that a logic gate only consumes current during switching. Therefore, the power consumption of a microcontroller in run mode is almost linear against bus frequency. The HCS12 typical current consumption while running is 2mA per MHz. This microcontroller family also features a very flexible PLL (Phase Locked Loop) that allows for choosing a low-frequency quartz or ceramic resonator hence achieving low power consumption when PLL is OFF. One can stop the CPU execution and selectively disable some modules so that only the used modules are clocked. This is the idea of the WAIT mode on HCS12. For even lower power consumption, the bus clock can be routed to only the Real Time Interrupt timer (RTI) and not across the whole chip. This is called the pseudo-stop mode on HCS12. The ultimate low-power configuration is to shutdown the oscillator circuitry. We are then in stop mode. The remaining power consumption comes from leakage currents that are highly dependent on ambient temperature.

In stop mode, the only possible wake-up events are external to the MCU. As all the internal peripherals are not clocked, they cannot generate an interrupt, except the MSCAN controller, the ports of which feature a special circuitry allowing “bus activity”. In stop mode, the wake-up events are:

- Forced hardware Reset
- IRQ interrupt: high priority falling edge sensitive interrupt (maskable)
- XIRQ interrupt: non-maskable interrupt (not recommended – see note 1 of **Table 1**)
- Key Wake-up interrupt: rising or falling edge configurable interrupt, only available on certain I/O ports (maskable)
- CAN bus activity: wake-up interrupt (maskable). See application note AN2255 “MSCAN Low-Power Applications” for more details and recommendations.

In pseudo-stop mode, the oscillator is not stopped. It is used to clock the Real Time Interrupt (RTI) allowing cyclic wake-up capability.

In WAIT mode, any unmasked interrupt can wake-up the microcontroller.

Entering low power modes

Low-power modes on HCS12 are entered by executing a specific instruction:

- “STOP” for stop or pseudo-stop modes with the S bit in the condition codes register (CCR) clear.
- “WAI” for wait mode

In order to enter pseudo-stop mode, one must set PSTP=1 (CLKSEL register) prior to executing the STOP instruction. Otherwise if PSTP=0, stop mode is entered.

In order to save time upon wake-up, the STOP and WAI instructions prepare the CPU to be interrupted by stacking the CPU registers in advance. Directly after that, the appropriate clock signals are stopped. In both cases, it uses the address of the following instruction as the return address for the Interrupt Service Routine (ISR) that is called by the wake-up event.

Table 1 shows the different HCS12 low-power modes against wake-up capabilities.

Table 1. HCS12 Low Power Features

Modes	Description	Wake-up Capabilities					
		Reset, /IRQ, /XIRQ (1)	Key Wake-Up Port	CAN	Real Time Interrupt	Watch-Dog	Other
Stop	Oscillator is OFF	YES	YES	YES (on activity)	NO	NO	NO
Pseudo-stop	Oscillator is ON but most of the system clocks are stopped (2)	YES	YES	YES (on activity)	YES	YES	NO
Wait	Oscillator is ON and peripherals are clocked	YES	YES	YES (possible to get 1st message)	YES	YES	Upon any non-masked interrupt

1. Be careful while using XIRQ to wake-up the application. This input does not behave like others upon external event. Please refer to HCS12 Core User Guide and read the STOP and WAI instructions description carefully.
2. Current consumption values within the HCS12 Device User Guides refer to Colpitts oscillator configuration.

In all these modes, the contents of registers (i.e. peripheral, port configurations etc.) remain unchanged as long as the device is powered within its specification limits.

The wake up conditions for each peripheral module need to be set according to the application needs prior to executing the STOP or WAI instruction. For the MSCAN, this also implies that the MCU puts the CAN interface in sleep mode for instance.

Exit of low power modes

Upon exit of wait and pseudo-stop, the CPU fetches the interrupt vector associated with the wake-up event. Note that this does not apply to wake-up from XIRQ when X mask bit is set. In this case (XIRQ interrupt disabled), execution continues with the next instruction after STOP. The application re-starts almost immediately given that the oscillator is still running.

Upon exit from stop mode, the same process applies. However, there will be an additional delay caused by the oscillator start-up. This delay varies greatly depending on the oscillator configuration (Colpitts or Pierce) and whether the application uses a crystal or a resonator as clock reference. Please refer to the Clock and Reset Generation (CRG) Block User Guide for more information

about the oscillator. Timings are similar to the oscillator start-up at power-up of the microcontroller. The fastest oscillator start-up is obtained by using a resonator in the Pierce oscillator configuration.

Table 2 below shows a collection of typical start-up timings using different crystals or resonators. These timings are expected to be similar for all HCS12 derivatives.

Table 2. HCS12 Oscillator Start-up Timing

Oscillator Configuration	Type of Quartz or Resonator	TYPICAL start-up timing
Colpitts mode	4MHz Quartz + C=22pF	8.00ms
	16MHz Quartz + C=12pF	2.50ms
	4MHz Resonator + C=39pF	< 1.00ms
Pierce mode	4MHz Quartz + 22pF + 0 Ohm	0.70ms
	4MHz Quartz + 22pF + 1.8KOhm	2.00ms
	16MHz Quartz + 10pF + 0 Ohm	0.50ms
	16MHz Resonator + 22pF + 1.8KOhm	< 0.20ms

System Basis Chip (SBC)

The SBC devices existing today have the following part numbers: MC33389, MC33889, MC33989. The philosophy of an SBC chip is to integrate a set of functionalities that are required on the large majority of automotive applications:

- Voltage regulator with reset controller
- CAN physical layer
- External watchdog
- Contact monitoring using high voltage inputs (Lx)
- Switched battery voltage (Vbat)

The MC33389 version features 2 regulators on-chip while the MC33889 and MC33989 have one 5V regulator (V1) plus the control circuitry for a second one (V2). The second regulator can be switched on and off independently of the main regulator. An external ballast transistor needs to be connected to the control circuitry via the Vsup, V2 and V2 CTRL pins. This allows the user to size this transistor closer to the application needs. The control circuitry takes the first regulator output as reference and works in tracking mode. Having all these functionalities in one single chip simplifies the low-power management. The control of the SBC operation is done via the SPI. Low power modes are entered by sending specific commands through the SPI.

The Freescale MC33889 (SBC with CAN LS) and MC33989 (SBC with CAN HS) devices have the same state-machine, meaning similar low-power modes. Apart from the different CAN physical interfaces, these two devices have slight differences in their feature sets (e.g. number of wake-up inputs…) but low power management is the same for both. This should also be the case for future versions.

Figure 1 shows the block diagram of the MC33989. The gray arrows highlight the possible wake-up events. They point at the pins that play a role in the wake-up detection and process. For consistency, the same symbolic representation of wake-up events will be used for all the figures within this application note.

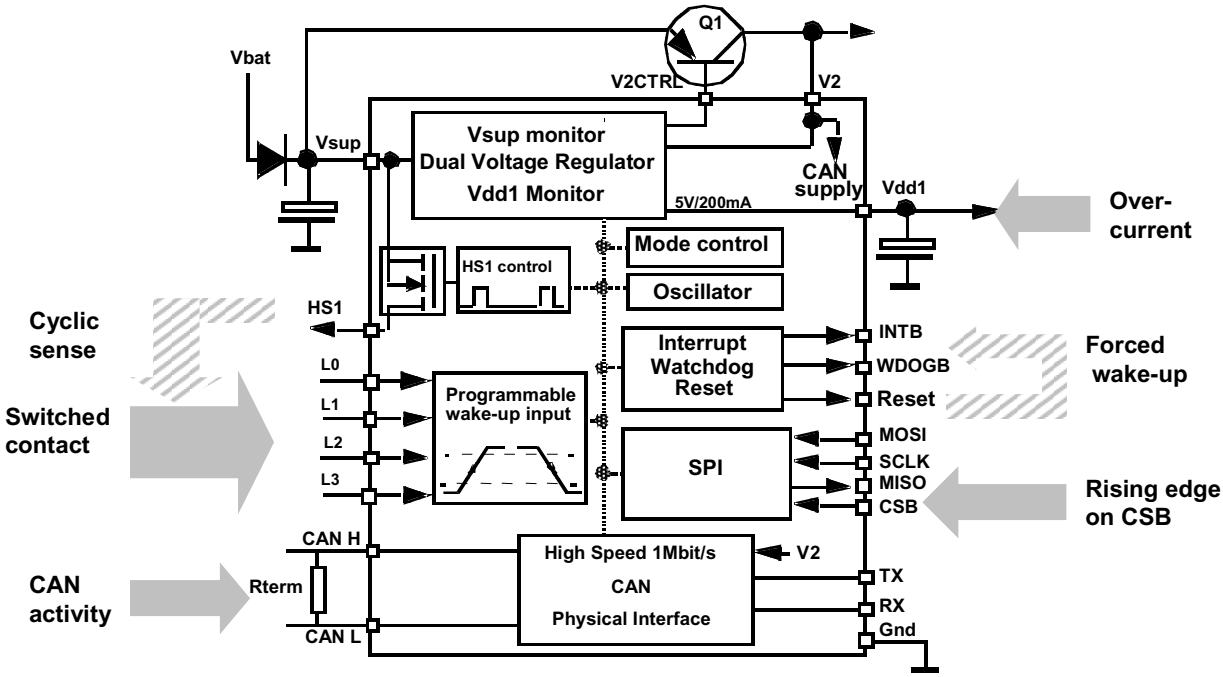


Figure 1. MC33989 Block Diagram

The primary wake-up event is a contact that is switched to battery voltage (Vbat) or ground (Gnd). Depending on the SBC implemented, there are up to 4 high-voltage inputs that can be used as wake-up inputs (Lx). These high-voltage inputs are rising or falling edge configurable and can be used as a port expander to read contact status once the application is running. Moreover, they can also be used for 5V signals due to their low switching thresholds.

There is the possibility to bias the wake-up inputs with the integrated high-side switch (HS1 output). HS1 can be switched ON and OFF cyclically by the internal oscillator. The switches (or contacts) can be supplied momentarily, hence reducing the quiescent current in low power modes. This feature is called “cyclic sense”.

The internal oscillator can be used alternatively to wake-up automatically after a pre-determined time spent in sleep or stop mode. This feature is called “Forced wake-up”. It can be seen as a wake-up upon timeout. Note that “cyclic sense” and “forced wake-up” functionalities are exclusive as they share the same timer.

The SBC can be woken-up by “CAN activity”. Please refer to product specifications for more details on voltage thresholds and patterns that trigger such a wake-up. Note that the CAN wake up circuitry is powered by an internal 5V regulator while in “sleep-enable” mode, leading to an additional 55µA typical current consumption. The CAN module itself is powered by V2 while enabled.

When the SBC detects a wake-up event, it generates a pulse on INTB in order to wake-up the MCU.

In the event that the MCU wakes-up before the SBC, it can wake-up the SBC in two ways:

- Applying a low to high transition on CSB.
- By pulling more current on Vdd1, thus triggering an over-current on Vdd1 output (over-current thresholds are lower in low-power modes).

Freescale recommends that the MCU toggles CSB in any case, even if a Vdd1 over-current wake-up has occurred. In some cases with very low oscillator frequencies, the MCU may not pull enough current to reach the over-current threshold and thus not wake-up the SBC. This functionality will be further explained in the next section of this application note.

Table 3 shows the different low-power modes against wake-up capabilities for MC33889 and MC33989.

Table 3. MC33889 or MC33989 Low Power Features

Modes	Description	Wake-up Capabilities					
		CAN (3)	Wake-up inputs	Cyclic sense (4)	Forced wake-up (4)	CSB (of SPI)	Vdd1 over-current
Sleep2	Vdd1 and V2 OFF	YES	YES	NO	NO	NO (MCU is not powered)	NO (MCU is not powered)
Sleep1	Vdd1 and V2 OFF internal osc. running	YES	YES	YES	YES	NO (MCU is not powered)	NO (MCU is not powered)
Stop2	Vdd1 ON V2 OFF	YES	YES	NO	NO	YES	YES
Stop1	Vdd1 ON V2 OFF internal osc. running	YES	YES	YES	YES	YES	YES

3. CAN needs to be in sleep-enable state to allow wake-up. This adds a typical 55µA current consumption
4. Cyclic sense and Forced wake-up are exclusive. They use the same timer, so if one is enabled, the other one cannot be enabled

The watchdog feature can be enabled if the oscillator is running (in sleep1 and stop1 modes). Thus, in the event that the MCU does not wake-up by itself and refresh the watchdog (via SPI) prior to its time-out, the SBC will activate its WDOG output to reset the MCU. This is a key feature preventing the application from getting stuck in “low-power mode” for any reason.

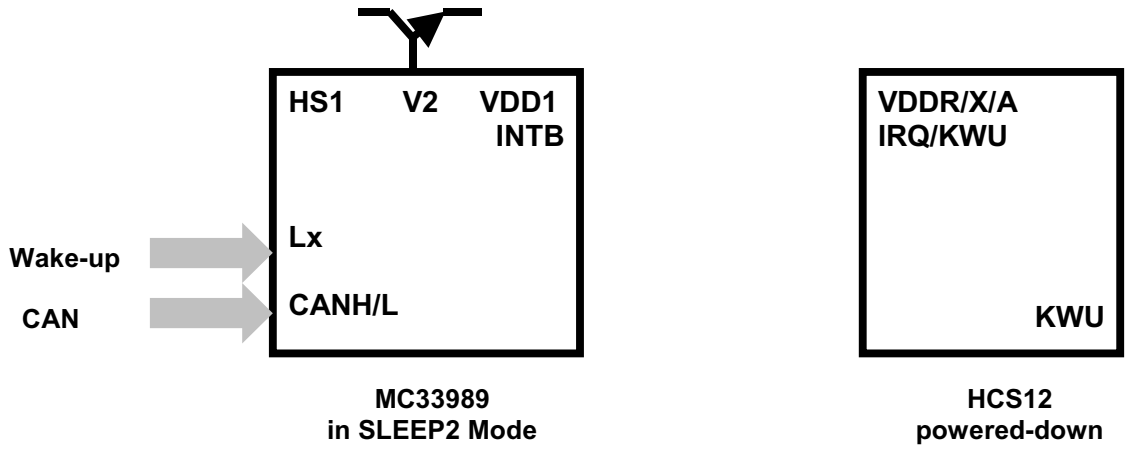
Low power strategies

The choice of the low-power strategy for the complete system is always a trade-off between the target power consumption, the wake-up capabilities and the wake-up timing. We can split the different strategies into two classes depending on the MCU being powered or not.

MCU not powered

The SBC can manage several different wake-up events, thus allowing the MCU to be unpowered in low-power modes. On the other hand, the wake-up timing is a bit longer, taking into account that the SBC should first wake-up itself, then have Vdd1=5V stable before releasing the reset. Then one needs to take into account the time needed for the oscillator of the MCU to start-up. For fast application start-up from power-down or stop mode, Freescale recommends configuring the HCS12 oscillator in “Pierce mode”.

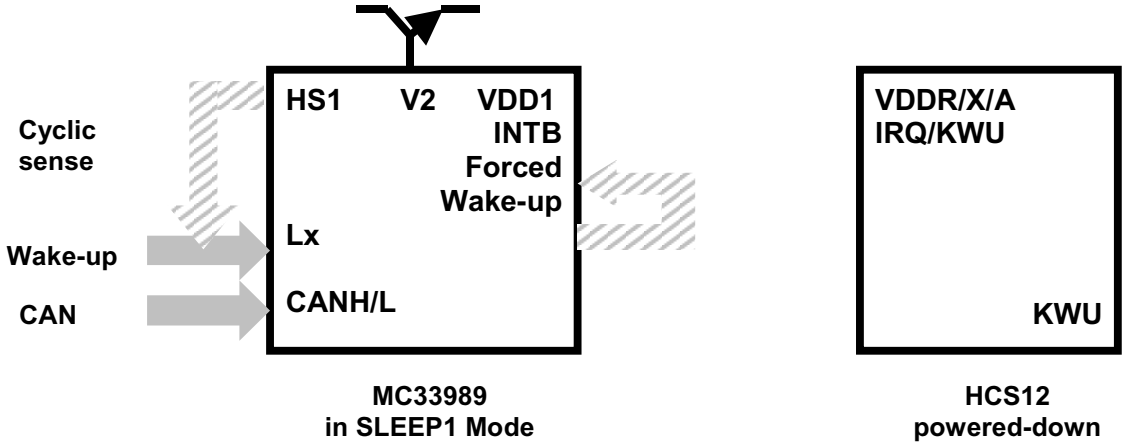
Figure 2 shows the most optimized power consumption allowing the bare minimum wake-up capabilities that are required in an automotive application, i.e. CAN activity and contact event. This assumes that the Wake-up circuitry does not add current consumption in idle state.



SBC Mode	TYPICAL Current Consumption	MAXIMUM Current Consumption
SLEEP2	57 + 55 (CAN sleep-enable) = 112µA	90 + 70 (CAN sleep-enable) = 160µA

Figure 2. SBC in SLEEP2 Mode

If cyclic wake-up is required, the internal oscillator can generate a time-out that wakes up the SBC. This functionality is called “Forced wake-up”. Alternatively, this oscillator can be used to cyclically power on and off the contacts or sensors used for wake-up events through HS1. This feature is called “Cyclic sense”. Note that the accuracy of the oscillator is +/-30%. Dashed arrows on **Figure 3** highlight the exclusivity between these two wake-up functionalities.



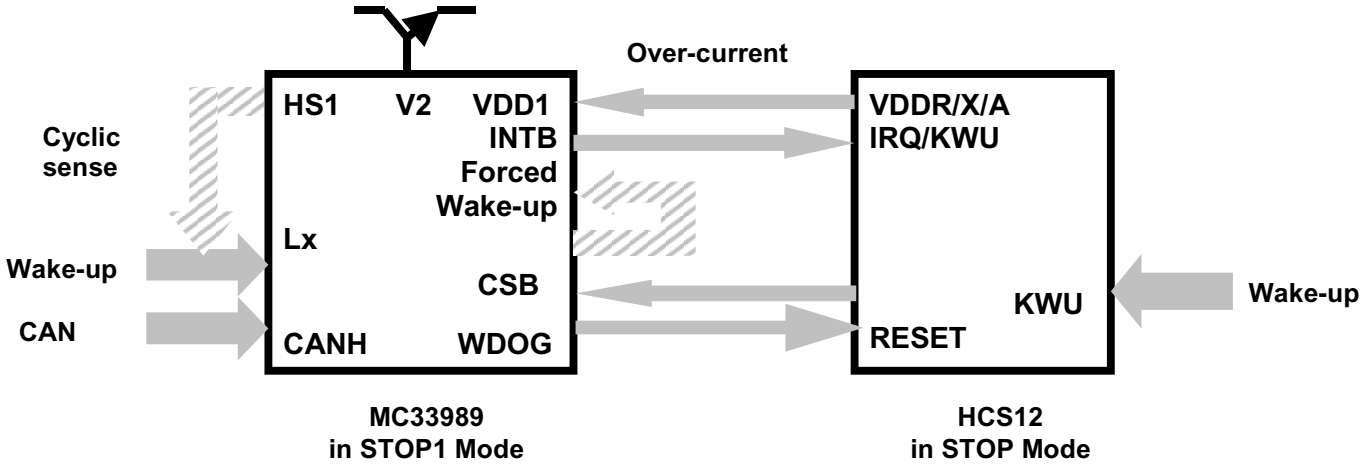
SBC Mode	TYPICAL Current Consumption	MAXIMUM Current Consumption
SLEEP1	72 + 55 (CAN sleep-enable) = 127µA	105 + 70 (CAN sleep-enable) = 170µA

Figure 3. SBC in SLEEP1 Mode

MCU powered

Powering the MCU may imply that the MCU manages the wake-up events alone and that we don't care which power supply is used. But this is not true with the SBC. The power supply still plays an important role, as it could also run in low-power model, thus managing a part or even all the wake-up events. The MCU can simply be powered to allow a faster recovery time.

Figure 4 shows the interactions between the HCS12 and SBC in addition to the wake-up capabilities that each device features.



SBC Mode	HCS12 Mode	TYPICAL Current Consumption	MAXIMUM Current Consumption
STOP1	STOP	135 + 55 + 25 (at T=25°C) = 215µA	Maximum value should be relative to a worst-case temperature profile

Figure 4. SBC in STOP1 mode + HCS12 in STOP

The MCU should set the SBC to stop1 mode shortly before entering its low power mode. The SBC features a built-in delay between the last rising edge of the CSB signal and the actual activation of stop1 (or stop2) mode. Then the SBC has a limited maximum output current on Vdd1. This maximum output current is high enough to cope with the large majority of low-power modes of any MCU. When the MCU needs more current, the SBC will automatically wake-up from stop1 mode and supply the required current. This is called an "over-current wake-up". This feature is a kind of insurance that the MCU will be powered properly upon exit from its low-power mode. If the MCU uses a low-frequency quartz, its current consumption may not be sufficient to wake-up the SBC. This is why Freescale recommends toggling the CSB signal in order to make sure that the SBC is up and running after the wake-up of the MCU occurred.

The above example with the "over-current wake-up" demonstrates how the SBC-HCS12 pair deals with the relative complexity of a system in which the

wake-up events are shared between two devices. In such a configuration we get:

- Wake-up on contacts or sensors that can be cyclically powered by HS1.
- Wake-up possibility from both sides (SBC or HCS12).
- Cyclic wake-up by the SBC (at +/- 30% timing accuracy).
- Faster wake-up since the MCU is already powered.

Another reason for powering the HCS12 is the much better accuracy of its cyclic wake-up (done by the Real Time Interrupt) in pseudo-stop mode versus the accuracy of the SBC internal oscillator in stop or sleep mode. If the HCS12 is in pseudo-stop, we still have the same configuration as on **Figure 3** with the additional functionality of cyclic wake-up via RTI on the MCU side.

We could also imagine interim low-power modes where some HCS12 peripherals are still running while the CPU has been put in WAIT mode. There are many different reasons for having a timer or a communication interface still running for a defined time period before the ECU goes into a “deeper” low-power mode. In case anything happens during this defined time period, the ECU can still react very quickly.

Table 4 below shows typical values of power consumption for an HCS12 in different low-power configurations at 25°C using a 4MHz quartz. Please refer to individual product specifications for accurate TYPICAL and MAXIMUM values.

Table 4. Indicative HCS12 Current Consumption

HCS12 CPU Mode + options	TYPICAL current 1)
RUN (PLL off, Fbus = 2MHz)	4 mA
WAIT + PLL & all peripherals enabled	3.7 mA
WAIT + ECT enabled	1.3 mA
WAIT (all modules disabled)	1.2 mA
PSEUDO-STOP (RTI enabled)	0.6 mA
STOP (oscillator is shutdown)	25µA

1.Note that some of these values are not product specifications, thus cannot be guaranteed by Freescale.

Example

Making the following reasonable assumptions:

- 2% of the time in RUN mode with a 4MHz quartz (meaning 2MHz bus frequency)
- A large part of the application that is powered by V2 (switched 5V) and HS1 (switched Vbat) is switched off in low-power

From **Table 4**, we can easily calculate the average current consumption in an application that needs to wake-up periodically to scan some I/Os and maybe do some A/D conversions.

Assuming the CPU spends 2% of its time in RUN mode at 2MHz bus frequency and the rest in pseudo-stop mode, then we get (using typical values taken from product specifications):

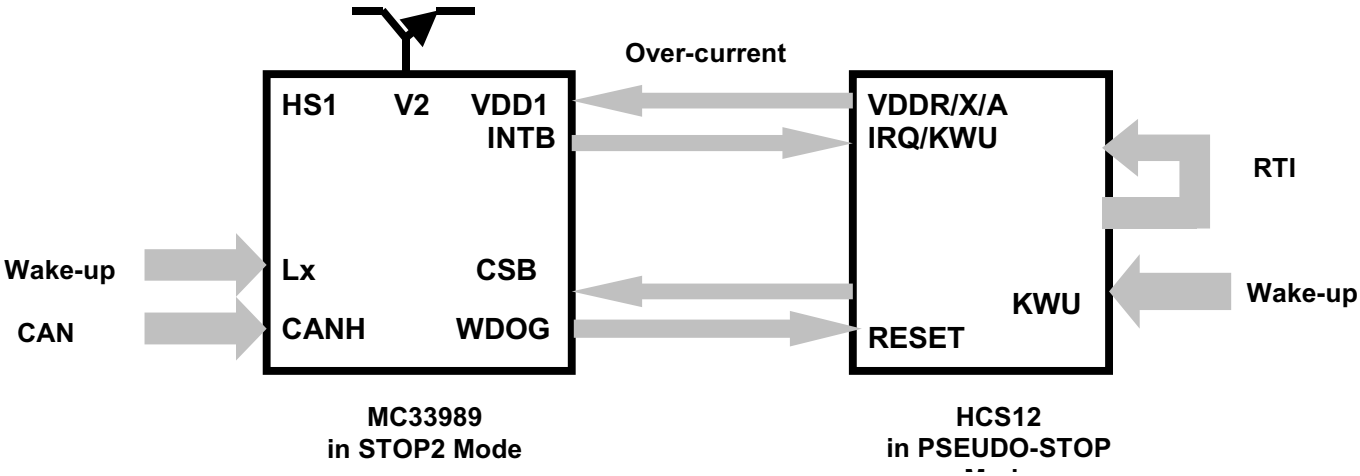
$$I_{average(HCS12)} = I_{run} \times 0.02 + I_{pseudo\text{-}stop} \times 0.98 = 4x0.02 + 0.6x0.98 = 0.670 \text{ mA}$$

The same type of calculation applies to the SBC (MC33989 taken for instance):

$$I_{average(SBC)} = (I_{sup} - I_{vdd1}) \times 0.02 + I_{stop2} \times 0.98 = 2.5 \times 0.02 + 0.130 \times 0.98 = 0.180 \text{ mA}$$

$$\text{Hence } I_{average} = I_{average(HCS12)} + I_{average(SBC)} = 0.850 \text{ mA}$$

Figure 5 illustrates this example, assuming that the HS1 and V2 power supplies remain OFF until the application comes back in RUN mode.



SBC Mode	HCS12 Mode	TYPICAL Current Consumption	MAXIMUM Current Consumption
STOP2	PSEUDO-STOP / RUN	Average 850µA	Depends on time spent in RUN vs PSEUDO-STOP

Figure 5. SBC in STOP2 mode + HCS12 in PSEUDO-STOP/RUN

Other considerations

I/O Configuration

When the HCS12 goes in any low-power mode, the content of its registers remain unchanged. In particular the ports keep their configuration. So, it is important to set the ports in a state that may not lead to a current consumption increase at the application level. Software and hardware engineers should follow these guidelines in order to avoid additional current consumption:

- One should not leave any I/O configured as an unconnected input but tie them to Vdd or Vss. Or one can also set unconnected I/O as output thus forcing a steady level.
- The same recommendation applies to unbonded I/O on small packages (on the QFP80 package vs QFP112 for instance). In this case we recommend setting the unbonded I/O as output.
- For inputs whose logic state is uncertain (for a Hall-effect sensor signal for instance), one should use external pull-up or pull-down resistors instead of the internal ones that are “weak” (typically between 20kΩ and 50kΩ). This way the power consumption is minimized in case the level of these inputs changes during the low-power mode.
  - **Example:** When using an input with internal pull-up, if the input signal goes low, we may get an additional current consumption of 5V / 20,000 = 250µA !
    Disabling the internal pull-up and using an external 100kΩ pull-up resistor in this case would only lead to 50µA additional consumption at the application level.
- In some cases, peripheral devices can be powered down (if connected to V2 of the SBC for instance). This implies that their I/O have an undefined state. In such cases it might be better to set the MCU port as an output in order to set a fixed level (to be defined depending on the device), thus avoiding floating nodes.

Temperature influence

It is well known that the operating temperature is one of the worst enemies of a semiconductor device. As we go down the CMOS technology curve, the leakage currents are getting higher. At very low power modes like stop or pseudo-stop on HCS12s, the temperature is the most influential parameter. Looking at the HCS12 specifications, the maximum power consumption values in these modes at high temperatures are quite far from their typical values at 25°C. So, special care should be taken at the start of the design phase to not “over-specify” the ECU but rather take a realistic worst-case temperature profile to assess which low-power mode to use on the HCS12. This generally drives the choice to power or not power the MCU and let the SBC manage the wake-up events alone.

Electromagnetic Immunity (EMI)

In order to protect the system from any unwanted wake-up, the system must feature good EMI performance. To help ensure that noises or glitches do not initiate the wake-up process, the SBC and HCS12 have built-in glitch filters. See individual product specifications for accurate information about glitches that are filtered on wake-up inputs and patterns required on CAN for wake-up.

Summary

This application note shows that the integration of key functions like voltage regulator and physical interface can bring more benefits than simply having both functionalities on one single chip. There are lots of additional features in terms of power management and safety on the SBC that cannot be implemented at low-cost with discrete components. Then if these smart features are coupled with the adequate low-power mode of the HCS12, the car-maker's requirements can easily be fulfilled. It also offers a flexible solution to Original Equipment Manufacturers (OEM) that have a platform approach where the needs and functionalities can be different from one application to another.

[redacted]

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