June, 2010

Low Power System Techniques

FTF-AUT-F0408

Carl Culshaw
System Architect
The lecture will:

- Focus on key low power challenges for the automotive system designer
- Describe the principal components of a microcontroller that help minimize power consumption
- Describe techniques to optimize their usage
- Discuss system partitioning approaches with a view to further reducing power
- Provide references and concepts to address the challenges identified

Presenter: Carl Culshaw

- System Architect for Automotive Microcontroller
- Experience: Automotive Body, Low Power; 8, 16, and 32-bit
Session Objectives

The lecture shall provide means to:

• Identify the key microcontroller elements to implement low power strategies
• Adopt technical approaches for their optimum usage
• Make the system architect aware of low power developments within the Automotive arena
Agenda

► Low Power Challenges
► Microcontroller Low Power Mechanisms
► Low Power System Techniques
► Summary
Low Power Challenges
Power Trend

Source: IRTS 2005 Power Consumption Trends for Soc-PE
As technology nodes increase more digital IP will be integrated but ....
Semiconductor Technology Trends
Source: TSMC 2009

...the impact of Performance vs. Power trade-off is not to be underestimated.
Power

- Dynamic Power – MCU Run Current
- Static Power – MCU Stop Current

\[ P_{\text{TOTAL}} = P_{\text{DYNAMIC}} + P_{\text{STATIC}} \]

Where

\[ P_{\text{DYNAMIC}} = \alpha \cdot CV^2f \]
\[ P_{\text{STATIC}} = I_{\text{STATIC}} V \]
Dynamic versus Static Mode Consumption

► When an MCU is running

\[ P_{\text{DYNAMIC}} = \alpha \cdot CV^2f \]

• Its transistors are switching
• Current is consumed at the point where the transistor switches.
  ▪ For example, in a CMOS inverter, at the point of switching both are on and an instantaneous current is drawn (crowbar effect)
  ▪ CMOS circuits dissipate power by charging the various load capacitances when switched.
• The more often the transistor switches, the more current is consumed.
  **Dynamic current: Typically linear with operating frequency**

► When an MCU is stopped

• Current is consumed due to leakage through the transistors.
• Sub-threshold/Channel Leakage:
  ▪ Occurs when the transistor gate is below the “on” threshold but a small current leaks from drain to source
• Becomes worse with reduced transistor size & higher temperature
• Drain to substrate leakage:
  ▪ The drain forms a diode with the substrate and there is a small leakage through this diode.
  **Static current: Typically exponential with temperature**
The Low Power challenge:

- Number of Electronic Control Units (mostly MCUs) is increasing exponentially.
  - Nodes requiring MCU functionality are replacing passive and mechanical systems throughout the vehicle.
  - Nodes include measurement points, actuation points, or control.
  - Often the vehicle architecture distributes some of the processing to local distributed nodes.
- Power consumption requirement is either flat or decreasing.
- Power allowed per Electronic Control Unit (ECU) is decreasing.

### Increase in number of ECU’s

<table>
<thead>
<tr>
<th>Golf I</th>
<th>Golf II</th>
<th>Golf III</th>
<th>Golf IV</th>
<th>Golf V</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>11</td>
<td>18</td>
<td>35</td>
</tr>
</tbody>
</table>

### Total Sleep current requirement

<table>
<thead>
<tr>
<th>Golf I</th>
<th>Golf II</th>
<th>Golf III</th>
<th>Golf IV</th>
<th>Golf V</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3mA</td>
<td>10.5mA</td>
<td>12.2mA</td>
<td>21.3mA</td>
<td>15.4mA</td>
</tr>
</tbody>
</table>

Example:
- Windows & Mirrors ................. 14
- Security and Access .............. 11
- Comfort & Information ............ 18
- Lighting .......................... 22
- Total ..................... 65

*Source: J. Leohold, VW, 9th International Automobile Electronics Conference, June 05*
Microcontroller
Low Power Mechanisms
Reducing Leakage – Active Well Bias

► Reduces leakage current in low power modes
► Changing the well voltage can reduce channel leakage by 10X. (Effectively, it increases $V_{th}$ for the transistors.)
► Applicable to standard cell logic and RAM arrays
State Retention Power Gating

► How it works:
  • Only D-Flip-Flops need to retain their state, all other logic can be powered down.
  • In D-Flip-Flops, only the state keeping latch needs to be powered.

► Special power rail is only for retention latches

► Example: S12X core
  • 15 µA consumption in state retention mode at 85C
Power Segmentation – MCU Structure

- **Power Domains**
  - Individually disconnected from Power
  - Eliminate leakage from areas that are turned off
- **Power Domain PD0:**
  - Always on
  - Wakeup periphery, e.g., CAN sampler, RTC, API, etc.
  - Minimum RAM segment
- **Power Domain RD1**
  - Contains an additional RAM segment
  - Remainder of the RAM is in the PD1 domain
- **Power Domain PD1:**
  - Contains all cores and the majority of peripherals
  - Can be turned off in STOP or STANDBY modes
  - Must be turned on in RUN modes

**MPC5604B Power Domain Structure**
Managing Clock Distribution

- Power consumption is directly linked to clock signal switching
- Reducing the number of clocked lines directly reduces current consumption
- Several methods are employed to avoid wasting power in clock edges, for example:
  - Clock freeze mode
    - When CPU activity can be temporarily halted, e.g., while waiting for an Analog-to-Digital (ADC) conversion to complete
  - Peripheral bus division
    - Reduced local clock rates, e.g., for ADC and communication ports
  - Clock gating
    - Applied wherever possible and at the entry to each sub-module

\[ P = a C_l V_{dd}^2 f \]
Intelligent Clocking – Typical Clock Sources

► Fast Wake-Up:
  • Increase the speed at which the device can recover from low power modes and start execution.
  • Reconfiguration and non-timing critical operations can start as soon as a clock is present.
    ▪ main RAM/module registers can be re-initialized whilst the external (accurate) OSC clock is stabilizing
  • Very fast Wake-Up requires an on-chip RC oscillator
    ▪ e.g., a 16 MHz Internal RC can provide Bus Clock in <5 cycles
  • Allows near instant operation

► Periodic Wake-Up:
  • Allows a reliable recovery in low power mode
  • Minimizes average current

<table>
<thead>
<tr>
<th>Clock Source</th>
<th>Start-up time</th>
<th>Power consumption</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>16MHz Internal RC Osc.</td>
<td>L (&lt;1us)</td>
<td>M (&lt;50uA)</td>
<td>L (&gt;10%)</td>
</tr>
<tr>
<td>32KHz / 128KHz Internal RC Osc.</td>
<td>L (&lt;1us)</td>
<td>L (&lt;1uA)</td>
<td>L (&gt;10%)</td>
</tr>
<tr>
<td>32KHz Ext. Crystal Osc.</td>
<td>H (ms)</td>
<td>L (&lt;10uA)</td>
<td>H (&lt;1%)</td>
</tr>
<tr>
<td>16MHz Ext. Crystal Osc.</td>
<td>H (ms)</td>
<td>H (&gt;100uA)</td>
<td>H (&lt;1%)</td>
</tr>
<tr>
<td>Internal PLL</td>
<td>H (ms)</td>
<td>H (&gt;1mA)</td>
<td>H (&lt;1%)</td>
</tr>
</tbody>
</table>
Dynamic Voltage Frequency Scaling (DVFS)

- Devices typically do not require full performance all of the time
  - Reducing frequency directly reduces power consumption (already discussed)
  - But reducing frequency also allows the voltage to be reduced, further reducing power

- Solutions can be via hardware, software or combination of both
  - System sophistication increases with DVFS
    - Typically PLL allows frequency scaling
  - Need to balance savings with complexity:
    - Scalable regulator design, with fast switching behaviour
    - Peripherals require full scaling capability to ensure seamless switching
    - Synchronization handling during DVFS changes

Note 1: Assumption is that maximum frequency also achievable at lowest voltage, but this will not always be the case
Low Power Modes

► STANDBY – lowest power mode
  • Power supply is cut off from most devices
  • All clocks disabled
  • Longest wakeup time and some reconfiguration required
  • Most pins not powered (high impedance)

► STOP
  • Advanced low-power mode during which the clock to the core and the PLL are disabled
  • Optionally switch off most peripherals
  • State of the output pins is kept

► HALT – disable core clock
  • Reduced-activity mode during which the clock to the core is disabled
  • Optionally switch off analog peripherals (PLL, flash, main regulator, etc.)

► RUN0-3
  • Software execution modes where most processing activity occurs.
  • Allows run-time customization of different clock & power configurations of the system
 Autonomous Operation

General theme is to switch on as little as possible:

- CPU is the most power hungry module on an MCU
- Need to switch to OFF when possible
- Put more intelligence into peripherals

Autonomous peripherals can help achieve this.

- Typical Autonomous peripherals include:
  - API – Autonomous Periodic Interrupt
    - Allows device to recover from very low power state at selectable time intervals
  - RTC – Real Time Clock
    - Offers time keeping functionality in very low power states
  - DMA – Direct Memory Access
    - Allows data transfer between peripherals minimizing CPU activity
  - ADC – Analog Digital Converter
    - Continual conversion while running in low power
    - Triggers wake-up when signal reaches certain level
  - LINFlex – Intelligent LIN management, minimizing CPU interrupts
Low Power System Techniques
Low Power System Techniques – Using the silicon

- System solutions to achieve low power are based on several key principles:
  - **Reduce average power**
    - Sleep as much as possible
    - Minimize RUN execution
    - Match speed against requirements

- **Only power what is needed**
  - Only switch on silicon portions
  - Completely power gate unused portions in many power modes

- **Only clock what is required**
  - Clock gating
  - Clock tree management
  - Peripheral grouping

- **Employ intelligent autonomous operation**

- **Design autonomous peripherals**
  - e.g. DMA, RTC, API, ADC, LINFlex
Energy Saving Potential on the System Level

Local, internal optimization of Electronic Control Units, actuators & sensors

Deactivation of complete sub-busses

Deactivation of individual ECUs

- Smart algorithms (PWM, motor control, and etc.)
- Loss minimization (DC/DC regulators, LEDs, and etc.)
- Smart activation of peripherals
- ...

- Network topology change
- Traditional wake-up mechanism
- Slow reaction times
- Limited SW changes

- Flexible wake-up mechanism required
- Slow reaction times
- SW and HW changes ➔ Industry standardization

Deactivation of individual ECUs

Savings Potential
Electronic Control Unit low power modes

- **SBC LPM** VDD OFF, no osc, no cyclic wake-up
  - MCU off
  - $I_{SBC} = 15\mu A$
  - Wake up time: 2-5ms + MCU S/W start

- **SBC LPM** VDDON with osc
  - MCU powered, STDBY mode no osc.
  - $I_{SBC} = 40\mu A$, $I_{MCU} = 30\mu A$
  - Very low system consumption
  - “longer” wake-up time

- **SBC LPM** VDD OFF, no osc, cyclic wake-up
  - MCU off
  - $I_{SBC} = 25\mu A$
  - Cyclic w/u allowed
  - “longer” wake-up time

- **SBC LPM** VDDON with osc
  - MCU powered, HALT mode with osc.
  - $I_{SBC} = 40\mu A$, $I_{MCU} < 350\mu A$
  - Faster w/u time
  - Impact on cons.
Example: Body Controller Module Typical Scenario

1. **MCU Mode: STANDBY**
   - + 8K RAM
   - + API/RTC (128K clock)
   - + Wake-up lines
   - Duration: ~667ms

2. **MCU Mode: RUN**
   - + Exec. from RAM
     - + IRC 16MHz
     - + Polling of ~24 digital + 5 analog inputs
   - Duration: ~200µs

3. **MCU Mode: RUN**
   - + Exec. from RAM
     - + XTAL for IRC trimming
     - + IRC 16MHz
     - + Polling of ~24 digital + 5 analog inputs
     - + SPI: 1x16 @ 500Kbps
     - + LIN: 3 msg at 20kbps
   - Duration: 32ms
Managing Communication Wake-Ups

High speed CAN

- SBC detects a wake up signal (3 dominant signals)
- Triggers a wake-up procedure.

- MCU Wakes up from an SBC signal.
Monitoring Fast Analog Signals

► Scenario:
  • Measure 3 ‘fast’ analog inputs and check values (e.g., ‘fast’ temperature sensors)
  • read the state of 5 port inputs and check values
  • Only continue full power-up if values breech predefined conditions

► Timing and absolute accuracy of initial measurements is not critical

► ADC function is active for approximately 9us (3us per conversion) in every 10ms

Proposed solution:

► Use approach: STANDBY → RUN → STANDBY → RUN

► Utilize STANDBY and retain 8K of RAM

► Utilize an API (Application Programming Interface) to wake-up periodically every 10ms and transition into RUN

► Clock the API with the on-chip 128KHz IRC (very low power Internal RC oscillator)

► Use a 16MHz IRC for fast execution, accurate enough for this example application

► Execute limited code from RAM at 16MHz bus speed
Monitoring Slow Analog Signals

**Scenario:**
- Measure 3 analog inputs and check values (for example, ‘slow’ temperature sensors)
- Sensors are **not** instantly available for reading
- Sensor settling time: 2ms

Timing and absolute accuracy of initial measurements is not critical

ADC function is active for approximately 9us (3us per conversion) in every 10ms

**Proposed solution:**
- This solution is identical as previous one, except the STOP state will be used instead of STANDBY during the ‘sensor stabilization’ period
  - Use approach: **STANDBY → RUN → STOP → STANDBY → RUN**
  - Return to **STANDBY** unless pre-defined conditions are exceeded
AUTOSAR 4.0 lacks specifications for ‘sleeping’ or functionally degraded ECUs.

Future AUTOSAR (AUTomotive Open System ARchitecture) enhancement:
- Support of partial networking (some kind of system degradation)
- Support of ECU degradation (implying MCU degradation)
- Dynamic degradation by System and ECU configuration

Partial networking requirements
- Flexible network management (Today: only synchronized fall asleep of all ECUs of one bus)
- Specification for configuration of wake-up of a single ECU (avoiding wake-up of all connected ECUs)
- Definition of configurable wake-up patterns
- Transceiver enhancement
- Configuration of partial networks and related signals

ECU degradation, a standardized approach for:
- De-initialization, re-initialization and re-configuration of software modules
- Adjustment on changed clock rates (of cores and/or other MCU blocks) which influences schedules (e.g., FlexRay).
- Adjustment on reduced clock accuracy which affects functions like PWM or CAN communication.
- Support of advanced low-power modes where some MCU blocks are still operative (e.g., PWM) while the rest of the MCU sleeps.
Session Summary

► Low Power Challenges
  • Automotive market demands for low power are increasing:
    ▪ More body nodes are appearing in every new generation of cars
    ▪ Each body node has more functionality and features
  • More functionality demands more from the silicon:
    ▪ Smaller geometries to meet increasing demands
    ▪ Smaller technology drives the power curve

► Microcontroller Low Power Mechanisms
  • Are there to make it easy
  • Software support

► System Low Power Techniques:
  • Power up fast
  • Standby as much as possible
  • Intelligent use of available resources

Attention to power is needed from the **beginning** of system design.

**Partitions, Power Down, Power States**
**Algorithm**
**System**
**Architecture**
**Logic Styles & Manipulation, Transistor Sizing, Energy recovery**
**Threshold Reduction, Double-Threshold Devices**

Technology

Accuracy

Power Savings