July, 2009

Migrating from 16-bit to 32-bit MCUs
For Automotive Applications

Steve McAslan
Applications Engineer, Microcontroller Solutions Group
Introduction

▶ The S12(X) and MPC56xx are independent processor families
  • Therefore, there is little “family resemblance”
  • However, the families target similar markets and use common approaches
    ▪ Interrupt architecture and low-power methodology are similar
    ▪ Development environment has similar tools and features
▶ Transferring hardware designs is generally not complex
  • I/O functionality is similar
  • Power supply is more flexible, and consumption is similar
▶ Porting software from one architecture to the other involves three classes of change
  • Architectural – 16- to 32-bit, endianism, reset and interrupt handling
  • Peripheral drivers – 32-bit family peripherals tend to be more advanced
  • Procedural – structure, computation and control
Processor Lineage
S12XE Compared to MPC5604C

**S12XEP100**
- 16-bit CPU
  - CISC architecture
- 16-bit RISC coprocessor (XGATE)
- Paged 23-bit memory space
- 1M byte on-chip flash
- 4k byte E-EEPROM (32k DFlash)
- 5 msCAN modules
- 8 SCI, 3 SPI, 2 I2C
- 32ch 12 bit ATD
- 24 8/16-bit timer (ECT, TIM, PWM)
- Interrupt controller with 8 priority levels
- Pierce oscillator 4- 16MHz
- FMPLL
- Expanded bus
- 50MHz/100MHz bus speed
- 112LQFP, 144LQFP, 208MAPBGA

**MPC5604C**
- 32-bit CPU
  - RISC with 16- & 32-bit opcodes
- Programmable DMA/CTU module
- Linear 32-bit memory space
- 512k byte on-chip flash
- 64k byte data flash
- 6 FlexCan modules
- 4 LINFlex, 3 DSPI, 1 I²C
- 28ch 10 bit ATD
- 28ch 16 bit eMIOS timer
- Interrupt controller with 16 priority levels
- Pierce oscillator 4- 16MHz
- FMPLL
- 64MHz bus speed
- 100LQFP, 144 LQFP
Comparing the User Mode Programming Models

User Programming Model

GPR0
GPR1
GPR31
LR
XER
CR
CTR

e200z0h

CPU12XE

XGATE

AccA | AccB
AccD
IX
IX
SP
PC
CCR

R0=0
R1 = Base
R2
R3
R4
R5
R6
R7
PC
CCR (N, V, C, Z)
MPC5604C CPU Programming Model (e200z0h)

**Processor Control Registers**
- SPR General Registers
  - SPRG0
  - SPRG1
- Machine State Register
  - MSR
- Processor Version
  - PVR
- Processor ID
  - PIR

**User Programming Model**
- User Programming Model
  - GPR0
  - GPR1
  - ... (continues)
  - GPR31
- LR
- CR
- XER
- CTR
- CSRR
  - 0
- DEAR
- IVPR
- SVR
- ESR
- CSRR
- IVPR
- CR
- XER

**Interrupt Register Set**
- Interrupt Register Set
  - DSRR0
  - DSRR1

**Debug Facilities Register Set**
- Debug Control
  - DBCR0
  - DBCR1
  - DBCR2
- Instruction Addr. Compare
  - IAC1
  - IAC2
  - IAC3
  - IAC4
- Debug Status
  - DBSR
- Data Address Compare
  - DAC1
  - DAC2

**Storage Control Register Set**
- Process Identification Register
  - PID0

**e200z0 Specific Registers**
- Data Value Compare
  - DVC1
  - DVC2

**e200z0 Book E Registers**
- Hardware Implementation Dependent Registers
  - HID0
  - HID1
- System Version
  - SVR
- Cache Config
  - L1CFG0
- BTB Control
  - BUCSR
- MMU Config
  - MMUCFG

---

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. 2009.
MPC5604C CPU Minus Interrupts and Power Modes

User Programming Model

- GPR0
- GPR1
- ...
- GPR31
- LR
- XER
- CR
- CTR

e200z0 Book E Registers

BTB Control

BUCSR
Hardware Considerations
S12, S12X and MPC560x devices feature on-chip voltage regulators to provide the power for the logic from a 3.3 – 5V supply

- The S12(X) regulator includes support for full-performance, low-power and stop modes
- The MPC560x provides three regulators for the same purpose
  - High-power, low-power, ultra-low-power regulators
- Some MPC560x derivatives can have independent voltages for different I/O banks

Devices with higher current consumption require an external “ballast transistor”
- Allows the extra heat to dissipate external to MCU

Power consumption is broadly similar between the S12XEP100 and MPC5604C
- Low-power modes on MPC560x are not static
The S12(X) and MPC560x devices use similar clocking structures:
- Pierce mode oscillator 4MHz to 16MHz
- Internal PLL with FM mode for full-speed operation
- Clock monitor feature to detect stable oscillators and loss of oscillation

The MPC560x MCUs provide additional clock sources including:
- Internal 16 MHz RC and separate 128 kHz RC oscillators
- External 32 kHz crystal oscillator

By default, the MPC560x devices always start on the fast internal RC:
- Crystal oscillator may optionally be started under software control
- PLL can operate on internal or external clock sources

In general, the MPC560x devices can operate multiple clock sources at the same time:
- Different peripheral sets can operate from different clock sources
I/O Considerations

► The MPC devices’ I/O pins have features that are similar to the S12(X) family
  • Most pins can be input or output
  • Most pins have the ability to select either pull ups or downs
  • Where required, there are dedicated pad types for special purposes such as ADC and stepper motor control
► I/O electrical characteristics are broadly similar
  • Input voltage thresholds, input leakage, pull-up/down values
  • Output voltage values are specified slightly differently
    ▪ Redesign I/O where absolute output V & I is important for a given load
► Mode and debug interfaces are different, but are shared with I/O
► For a hardware designer, the primary concern is to verify that a particular pin has the attributes required for the design (rather than having to redesign I/O circuitry)
Typical MCU Hardware Configuration

- **S12XE**
  - Mode & debug
  - I/O load
  - Other S12(X)s may require external PLL components
  - VDD decoupling

- **MPC560x**
  - Mode & debug
  - I/O load
  - Some MPC560x may require external VREG components
  - VDD decoupling

Reconfirm I/O design
The Development Environment
The MPC560x devices are supported by a range of tool chains from various suppliers, including some common to the S12X

- CodeWarrior, Cosmic
- Wind River/Diab and Green Hills offer comprehensive tool chains and experience with Power Architecture®

The on-chip debug support uses a different physical connection between the two families

- BDM/DBG vs Nexus

The basic development tool support operates in a similar way and from similar suppliers

- P&E, Lauterbach, iSystems
BDM vs Nexus

- MODA, MODB, MODC, and RESET are shared signals.
- Nexus signals required for trace functionality:
  - TCK
  - TDI
  - TDO
  - TMS

Optional additional Nexus signals required for trace functionality:

- FABM
- ABS
- ~8 Nexus signals

Internal Trace Buffer

6 pin header

14 pin header

38 pin MICTR

Shared Test Access Port (TAP)
BDM/DBG vs Nexus Functionality

► MPC560x devices include the industry-standard Nexus interface
  • Nexus 2+ includes all level 1 & 2 features and some from levels 3 & 4
  • Trace requires external buffer to capture trace data
► Common features between the interfaces are
  • Enter a debug mode from reset or user code, read/write user registers or memory in debug mode, single step instructions in user mode and re-enter debug mode
  • Set breakpoints or watchpoints, stop program execution on instruction/data breakpoint and enter debug mode
  • Device identification
  • Trace program flow in real time, read/write memory locations while processor runs in real-time
  • Start data or program traces upon watchpoint occurrence, ability to stall the processor when trace buffers are full
Software: Architectural Impact
Considerations for C Programmers

► Reuse of significant portions of procedural code is possible with some cautions
  • By default, integer type will change from 16- to 32-bit
    ▪ Review integer type ranges and opportunity to over/underflow
    ▪ Review bitwise operation values
  • By default, pointer native size will also change
    ▪ Review pointer arithmetic code
  • RISC assembly code will execute in a different time
    ▪ Review sensitivities to code execution time
  • Writes into registers may have unexpected effects if written with larger/smaller data types
    ▪ Optimizations may be possible by change variable type sizes
► The Freescale header files use a similar structure to existing S12(X) files (MODULE.REG.R, MODULE.REG.B.BITNAME)
► Although both architectures are big-endian, the bit numbering is different
  • S12X has bit 0 as the Isb and MPC560x has bit 0 as the msb
Power Consumption Considerations

- The operation of the families when switching between run and low power is different
  - On S12(X), each peripheral has control bits to control if it is active or not and control the behaviour of some peripherals in low-power modes
  - MPC560x devices use an automatic system to control the operating mode of the device

- The advantage of the MPC560x approach becomes evident when power mode switching requires that peripherals are individually enabled or disabled
  - A central “database” of the active peripherals in each mode allows the mode switch to be performed with a simple instruction sequence
  - The “database” must be configured for all of the valid modes before use
There are 10 different operating modes
- Plus a mode for RESET
- The user configures the behavior of each mode using the Mode Entry, Clock and Power control modules
- There are four “generic” RUN modes that can be used to optimise power consumption when the MCU is active
- There are three reduced power (“non-run”) modes that are used when the device is not actively executing application code
  - HALT, STOP, STANDBY (cf WAIT, PSTOP, STOP)
- There are three system modes that are used when configuring the system and recovering from errors
- Low-power modes are enabled in the MCSR
## S12X Interrupt Architecture

<table>
<thead>
<tr>
<th>Vector</th>
<th>Interrupt/Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFFE</td>
<td>POR</td>
</tr>
<tr>
<td>0xFFFFE</td>
<td>LVR</td>
</tr>
<tr>
<td>0xFFFFE</td>
<td>External</td>
</tr>
<tr>
<td>0xFFFFE</td>
<td>Illegal address</td>
</tr>
<tr>
<td>0xFFFC</td>
<td>CMU</td>
</tr>
<tr>
<td>0xFFFFA</td>
<td>COP</td>
</tr>
<tr>
<td>0xF8</td>
<td>Unimplemented instruction</td>
</tr>
<tr>
<td>0xF6</td>
<td>SWI</td>
</tr>
<tr>
<td>0xF4</td>
<td>XIRQ</td>
</tr>
<tr>
<td>0xF2</td>
<td>IRQ</td>
</tr>
<tr>
<td>0xF0</td>
<td>RTI</td>
</tr>
<tr>
<td>0xEE</td>
<td>ECT0</td>
</tr>
<tr>
<td>0xEC</td>
<td>ECT1</td>
</tr>
<tr>
<td>0x14</td>
<td>MPU</td>
</tr>
<tr>
<td>0x12</td>
<td>SYS</td>
</tr>
<tr>
<td>0x10</td>
<td>Spurious Interrupt</td>
</tr>
</tbody>
</table>
### S12X Interrupt Architecture

#### Vector and Interrupt/Reset

<table>
<thead>
<tr>
<th>Vector</th>
<th>Interrupt/Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFFFFE</td>
<td>POR</td>
</tr>
<tr>
<td>0xFFFFE</td>
<td>LVR</td>
</tr>
<tr>
<td>0xFFFFE</td>
<td>External</td>
</tr>
<tr>
<td>0xFFFFE</td>
<td>Illegal address</td>
</tr>
<tr>
<td>0xFFFFC</td>
<td>CMU</td>
</tr>
<tr>
<td>0xFFFFA</td>
<td>COP</td>
</tr>
<tr>
<td>0xF8</td>
<td>Unimplemented instruction</td>
</tr>
<tr>
<td>0xF6</td>
<td>SWI</td>
</tr>
<tr>
<td>0xF4</td>
<td>XIRQ</td>
</tr>
<tr>
<td>0xF2</td>
<td>IRQ</td>
</tr>
</tbody>
</table>

#### Interrupts handled by the interrupt controller via I bit

<table>
<thead>
<tr>
<th>Vector</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xF0</td>
<td>RTI</td>
</tr>
<tr>
<td>0xEE</td>
<td>ECT0</td>
</tr>
<tr>
<td>0xEC</td>
<td>ECT1</td>
</tr>
<tr>
<td>0x14</td>
<td>MPU</td>
</tr>
<tr>
<td>0x12</td>
<td>SYS</td>
</tr>
<tr>
<td>0x10</td>
<td>Spurious Interrupt</td>
</tr>
</tbody>
</table>

**Automatic handling**

INTC has 8 priority levels and 10 software vectors (including 8 from XGATE)
MPC560x Interrupt Architecture

<table>
<thead>
<tr>
<th>Vector</th>
<th>Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>IVOR0</td>
<td>Critical input</td>
</tr>
<tr>
<td>IVOR1</td>
<td>Machine Check</td>
</tr>
<tr>
<td>IVOR2</td>
<td>Data Storage</td>
</tr>
<tr>
<td>IVOR3</td>
<td>Instruction storage</td>
</tr>
<tr>
<td>IVOR4</td>
<td>External</td>
</tr>
<tr>
<td>IVOR5</td>
<td>Alignment</td>
</tr>
<tr>
<td>IVOR6</td>
<td>Unimplemented instruction</td>
</tr>
<tr>
<td>IVOR8</td>
<td>System call</td>
</tr>
<tr>
<td>IVOR15</td>
<td>Debug</td>
</tr>
</tbody>
</table>

Interrupts handled by the interrupt controller (INTC) via IVOR4

- 0x0800: Software flag 0
- 0x0804: Software flag 1
- 0x0808: Software flag 2
- 0xB40: FlexCAN_BUF_12_15
- 0xB44: FlexCAN_BUF_16_31
- 0xB48: FlexCAN_BUF_32_63

INTC has 16 priority levels and 8 software vectors
MPC560x Interrupt Operation

➤ Unlike the S12X, the MPC560x does not automatically stack registers on servicing an interrupt
  • Tool chains provide standard “prolog” and “epilog” code to do this
  • Depending on the interrupt handler, it may be possible to service a request more quickly by bypassing these standard functions

➤ The MPC560x also has a choice of how to handle the “external” interrupts managed by the interrupt controller (IVOR4)
  • In Hardware Vector Mode each interrupt has a unique vector entry containing the jump address of the ISR
    ▪ The software must initialize the INTC with the correct vector entries
  • In Software Vector Mode all interrupts go to the IVOR4 handler
    ▪ The handler identifies the actual interrupt source by reading a register then finds the location of the ISR from a jump table
  • Software vector mode is slower but allows the use of common prolog and epilogs

➤ The IVOR interrupts are enabled using bits in the CPU MSR register
Reset Architecture

- The S12X has fixed locations for its reset vectors and fixed functionality for dealing with reset sources.
- The MPC560x provides a choice of reset vectors and the option to allow the system to use a reset event as a non-maskable interrupt instead.
  - All resets share the same chosen vector and are distinguished by polling bits in a status register.
- There are multiple locations for the reset vector in internal flash.
  - The chosen vector must have a valid “Reset Configuration Half Word” value as part of the vector.
  - If no RCHW value is present, the vector at that location is ignored.
- When a reset event that is configured as an interrupt occurs, the MCU automatically jumps from its current mode to a new SAFE mode in which a known operating configuration is used (cf. CMU).
  - This includes use of IRC rather than crystal, internal flash is enabled.
Software: Peripheral Drivers
Peripheral Compatibility

► In general, there is limited compatibility between peripherals on the two families
  • In some cases, the original S12(X) peripheral is converted to a 32-bit version and is mostly code compatible
  • These include IIC, LCD, SMC, and SSD
► The MPC560x peripherals feature extra functionality over the S12(X) equivalents and driver will have to be modified to support the different peripheral
  • The MPC560x peripherals are often reused from the MPC5xxx family so application notes and driver software already exists
► The following slide lists the MPC560x peripheral that most closely matches that on the S12X
  • The MPC560x peripherals generally have support for system DMA functionality
### Peripheral Comparison

<table>
<thead>
<tr>
<th>S12X</th>
<th>MPC560x</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>msCAN</td>
<td>FlexCAN</td>
<td>Full message buffer supported</td>
</tr>
<tr>
<td>SCI</td>
<td>LINFlex</td>
<td>Buffered SCI with integrated LIN support</td>
</tr>
<tr>
<td>SPI</td>
<td>DSPI</td>
<td>Buffered SPI</td>
</tr>
<tr>
<td>ECT, TIM, PWM</td>
<td>eMIOS</td>
<td>Timer system with flexible architecture</td>
</tr>
<tr>
<td></td>
<td>eTimer</td>
<td>Motor control focused timer</td>
</tr>
<tr>
<td>PIM</td>
<td>SIU</td>
<td>Similar functionality, pin rather than port focus</td>
</tr>
<tr>
<td>EBI</td>
<td>QuadSPI</td>
<td>SPI system with four I/O lines</td>
</tr>
<tr>
<td>ATD</td>
<td>ATD</td>
<td>Programmable ATD with full channel flexibility</td>
</tr>
<tr>
<td>MMC</td>
<td>XBAR</td>
<td>Crossbar switch allowing simultaneous access to multiple masters</td>
</tr>
<tr>
<td>MPU</td>
<td>MPU</td>
<td>Similar design up to 16 descriptors, 32b granularity</td>
</tr>
<tr>
<td>XGATE</td>
<td>DMA</td>
<td>Programmable DMA</td>
</tr>
<tr>
<td></td>
<td>CTU</td>
<td>Cross-triggering unit allowed direct interaction between peripherals</td>
</tr>
</tbody>
</table>
Software: Structure, Computation & Control
Software Structure

► PRO: A significant benefit of moving to the MPC560x family is the 32-bit linear address range
  • All peripherals and memories are memory mapped into this 4 GByte space
  • The most likely impact for S12X users is the ability to reduce the dependency on linker settings and code pragmas

► CON: A potential disadvantage is the loss of the XGATE for simultaneous processing
  • It may be that the MPC560x peripherals themselves include the functionality provided by XGATE (for example buffers)
  • The DMA allows manipulation of the source and destination address for each transfer; it may be able to implement some basic data ordering functions
  • The Crossbar switch allows simultaneous access for masters such as the CPU and DMA
Summary
Summary

► Despite their different origins, the convergence of the MPC560x family and the S12(X) family means that migration of applications may not be as complex as expected
  • Similar hardware configuration and development tools
  • Similar power consumption profiles
► The configuration of the MPC560x is flexible enough to mimic the approach of the S12X for system level features, such as interrupts and low-power modes
  • With the added benefit of more configuration options and more systematic approaches
► Reuse of software for procedural and control purposes is highly feasible, although most peripheral software will have to be rewritten
  • The more advanced peripherals will be generally beneficial to the performance of the system
Thank you for attending this presentation. We’ll now take a few moments to review the audience questions, and then we’ll begin the question and answer session.