Freescale Semiconductor
Application Note

Document Number: AN4143
Rev. 1, 06/2010

Migrating from the 16-bit S12XE to 32-bit Qorivva MPC5604B Family of Microcontrollers

by: Alasdair Robertson
Murray Stewart
Steve McAslan
Steven McLaughlin

Applications Engineering
Microcontroller Solutions Group
East Kilbride, Scotland

Contents

1 Introduction

Freescale has introduced the Qorivva MPC5604B family of microcontrollers to provide cost-effective solutions for mid-range automotive body applications. The family provides higher performance than the existing industry standard S12 and S12X microcontroller families in similar configuration and package options and introduces Power Architecture™ to this application space. This application note is targeted at users who wish to migrate to the new family.

1.1 Migrating an application

Much of the effort migrating an application will be in gaining knowledge of the features of the Power Architecture and how they compare to the existing S12X features. This application note is arranged into specific

sections that present the MPC5604B family from the perspective of an S12X user for key architectural features. The remainder of this section presents an overview of the migration process starting with the MPC5604B families and the application areas they target.

1.1.1 MPC560xx families

The MPC560xx family nomenclature operates differently from that used by the S12X. The part numbers are created as follows.

There are three main families within the MPC560xx range:

- The processor CPU is identified using the digit after the MPC56 designation. A ‘0’ indicates that the CPU is an e200z0h variant of the Power Architecture family. The features of this CPU are described in the core manuals. Other higher performance devices are also available within the MPC56xx family but these are not discussed here.
- The digit following the CPU designation indicates the memory size, for example ‘4’ indicates 512 KB of flash, ‘6’ indicates 1 MB of flash.
- The letter following the memory size digit indicates a particular variant that is related to the feature set of the device. This is similar to the use of letters in the S12X family to indicate variants: D-family, S12XD; E-family, S12XE; etc.

The three families are denoted ‘B’ for general car body and gateway applications (MPC5604B), ‘P’ for chassis, steering, and other hydraulic applications (MPC5604P), and ‘S’ for dashboard cluster and driver information applications (MPC5606S). Broadly speaking, the MPC560xB and MPC560xP devices are targeted at similar applications as the S12XD and S12XE microcontrollers while the MPC560xS devices offer similar features as the S12XH families.

1.1.2 General software features

The peripherals and memory maps are mostly different between the two families. (There are some modules that have been reused with minor changes from the S12X family, and in these cases it may be possible to reuse some high-level software.) This means that software development will be the most significant challenge in migrating an application.

Even for software that can be reused, there are a number of important considerations that the developer must keep in mind

- The MPC5604B CPU is fully 32-bit with all addresses and data types normalizing to that size. This means that integer sizes change with a potential corresponding impact on overflow and underflow and bit manipulation. Pointer sizes also change.
- Execution time will be different due to the nature of the CPU architectures.
- Writing to peripheral registers with smaller than an integer data type may cause unexpected behavior.

Endianism is compatible between the parts but the nomenclature is different. One must remember that endianess describes how a processor interprets memory content. This is different from bit numbering notation. On the S12X the most significant bit always has the highest value — for example, the most significant bit of a 16-bit peripheral register is labelled 15, whereas on the MPC5604B the most significant

bit is always labelled 0. The most significant bit is still the left-most position in a register or memory location.

1.2 Architecture and convergence

Both the S12X and Power Architecture families are well-established in the automotive industry, but until now there has been limited overlap between the devices. The Power Architecture has traditionally dominated in higher performance applications such as engine and chassis control while the S12X has been most successful in power-sensitive areas such as car body.

Figure 1 illustrates how both families have evolved over the last two decades such that the devices are now converging on a common application area.

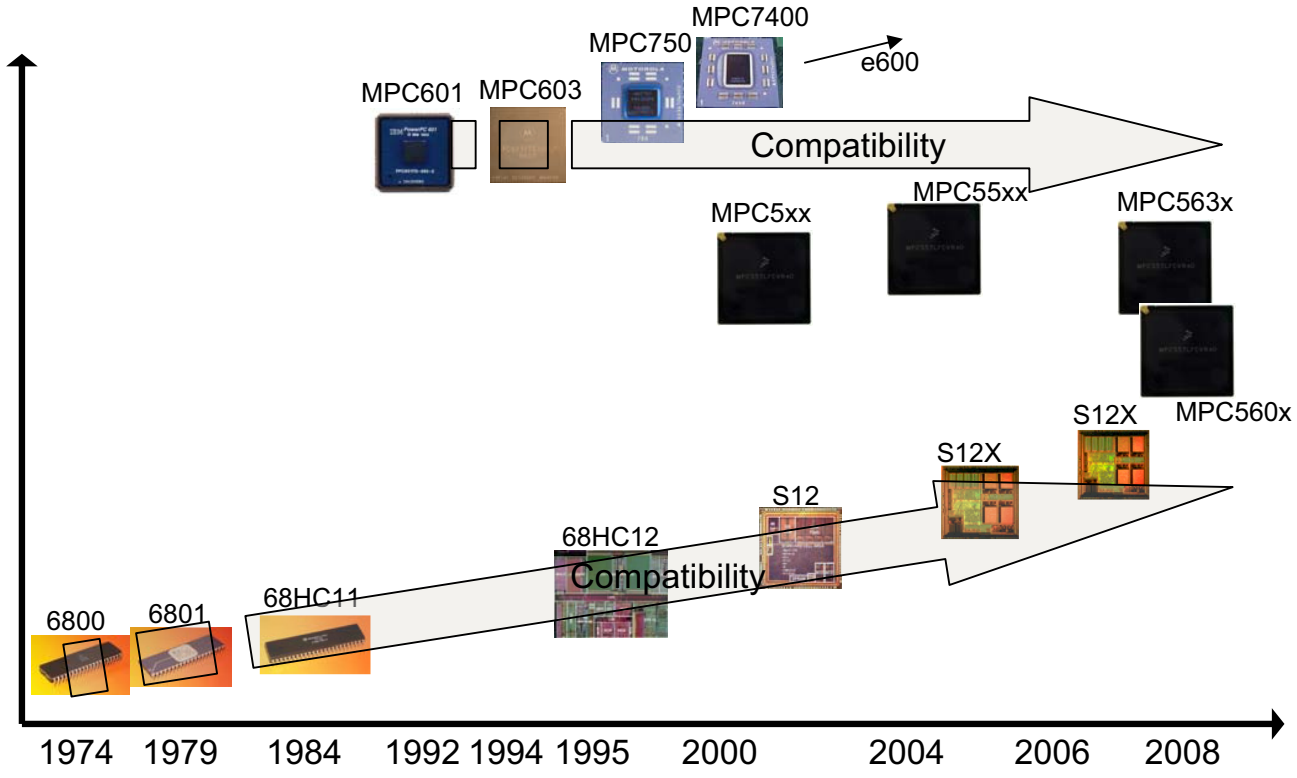

Figure 1. Architecture convergence

Note that compatibility has been maintained within both families between the current generation and previous ones. For S12X the primary change has been in improved performance while maintaining its suitability for the target application (for example, migrating from S12XD to S12XE), while for the Power Architecture devices the changes have allowed better optimization of performance and power consumption.

The net result has been to allow the Power Architecture and S12X families to converge in the mid-range car body application area. This fact is significant because the result of the convergence is that both families now share many common features while maintaining compatibility and consistency with previous

generations. The convergence allows customers familiar with the S12X family to continue using familiar package options, power supply designs, PCB layout experience, and in some cases software, to implement solutions with all the required features plus higher performance.

Table 1 contains a summary of the feature set of the MPC5604B and the S12XEQ512 devices

Table 1. Comparison of S12XE and MPC5604B microcontrollers

Feature	S12XEQ512	MPC5604B
CPU	16-bit CISC S12X	32-bit RISC Power Architecture with 16- and 32-bit opcodes
Co-processor/DMA	XGATE, RISC core	16-channel DMA plus Cross-Triggering Unit[1]
Memory map	Paged 23-bit	Linear 32-bit
Flash	512 KB	512 KB
EEPROM/Data flash	4 KB / 32 KB	32 KB
RAM	32 KB	48 KB
CAN	5 × MSCAN	3 × FlexCAN
Serial communications	8 × SCI, 3 × SPI, 2 × I^2C	4 × LINFlex, 3 × DSPI, 1 × I2C
ATD	16 × 12-bit	28 × 10-bit[2]
Timer channels	24 (ECT/TIM/PWM)	28 (eMIOS)
Interrupts	8 priority levels	16 priority levels
Oscillator	Pierce 4–16 MHz	External: Pierce 4–16 MHz (FXOSC) / 32 kHz (SXOSC). Internal: 16 MHz (FIRC), 128 kHz (SIRC)
PLL	PLL with FM	PLL with FM
Expansion	Expanded bus	n/a
Bus speed	50 MHz	64 MHz
Packages	112 LQFP, 144 LQFP	100 LQFP, 144 LQFP, 208 BGA (Nexus development only)

1 DMA is not present on MPC5604B but on MPC5602D and MPC5605/6/7B.

2 ATD has 12-bit resolution on MPC5607B.

2 Architectural differences

2.1 CPU features

Since the families have been developed independently from each other there is no compatibility between the CPUs of the S12X and the MPC560xx. The MPC5604B is fully Power Architecture compliant and hence is a RISC processor with thirty-two general purpose registers and the ability to execute opcodes in a single cycle. Additionally, the e200z0h CPU is implemented using Harvard architecture (the ‘h’ on e200z0**h** denotes Harvard architecture). Users of the S12X family will be familiar with the features of

RISC processors through contact with the XGATE and experience in implementation of software in assembly (directly, or from high-level language) continues to be relevant.

Figure 2 shows the full programmer's model of the e200z0h CPU. Static features are present for compatibility reasons but perform no configuration function on this CPU. These are highlighted in blue and can be ignored for the purposes of application development. On the S12X family, debugging features are provided by the DBG module. On the e200zh, almost identical features are located in the CPU itself and are highlighted in green. In both cases the configuration and use of these features is performed by debugging tools and therefore can be ignored for most software development purposes. The e200z0h takes a different approach to handling interrupts, and this requires dedicated registers in the CPU. In most cases a common interrupt handling algorithm is used to manage this process, and so these registers (highlighted in red) can be disregarded for general software development. The remaining registers are either general purpose in nature or, in the case of the BTB (branch optimization settings), are used to configure the CPU performance for a particular software configuration.

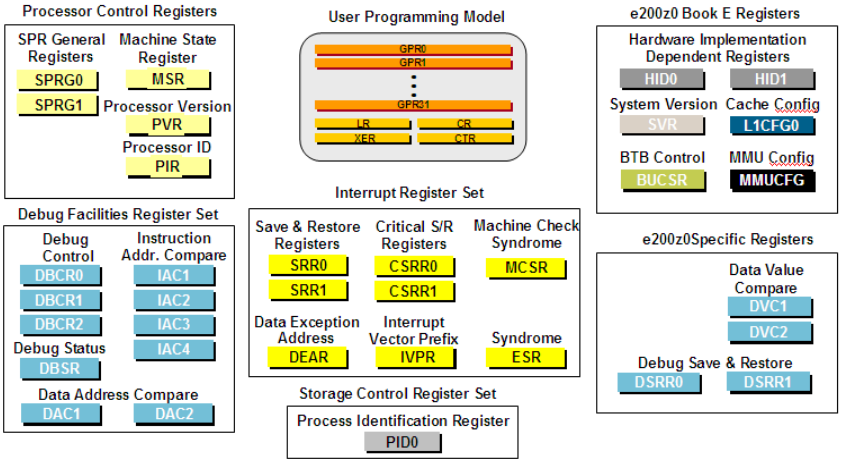

Figure 2. e200z0h programmers model

2.2 Peripherals

As a general rule the peripherals are different on the MPC5604B and so new drivers and software handlers will be needed. However, some peripherals are re-used from the S12X (see specific MCU information for details) and most others are re-used from other MPC55xx devices. This means that many peripherals will already have drivers available that can be used immediately, and there is example application software published in application notes.

Table 2 highlights the various modules which were available on S12XE and the equivalent module on the 32-bit MPC5604B.

Table 2. Module-by-module comparison of MPC5604B vs. S12XEQ512

S12XE	MPC5604B
Analog-to-digital convertor (ADC)	Analog-to-digital convertor (ADC)
N/A	Boot Assist Module (BAM)
N/A	CANSAMPLER
Clocks and Reset Generator (CRG)	Clock Generation Module (CGM)
Clocks and Reset Generator (CRG)	Clock Monitor Unit (CMU)
X-GATE	Cross Triggering Unit (CTU)
Serial Peripheral Interface (SPI)	Deserial Serial Peripheral Interface (DSPI)
CPU12	e200z0h
Flash	Error Correction Status Module (ECSM)
Enhanced capture timer/Timer (ECT or TIM)	Enhanced Modular I/O Subsystem (eMIOS)
Freescale's Scalable Controller Area Network (MSCAN)	FlexCAN
FMPLL	FMPLL
I^2C	I^2C
Interrupt Controller (INTC)	Interrupt Controller (INTC)
Background debug mode (BDM)	IEEE 1149.1 test Access Port Controller (JTAG)
Background debug mode/Debug (BDM/DBG)	Nexus Development Interface (NEXUS)
Debug Module (DBG)	N/A
Serial Communication Interface (SCI) (can emulate LIN)	LIN Controller (LINFlex) — with SCI/UART
N/A	Mode Entry (ME)
Memory Protection Unit (MPU)	Memory Protection Unit (MPU)
External Bus Interface EBI	N/A
N/A	Power Control Unit (PCU)
Periodic Interrupt timer (PIT)	Periodic Interrupt timer (PIT)
Clocks and Reset Generator (CRG)	Reset Generation Module (RGM)
N/A	Register protection (RPM)
Port Integration Module (PIM)	System Integration Unit (SIU)
FLASH	System Status and Configuration Module (SSCM)
Clocks and Reset Generator (CRG — COPCTL register)	System Watchdog Timer (SWT)
At peripheral	Wakeup Unit (WKUP)
Memory Mapping Control (MMC)	Crossbar Switch (XBAR)
Enhanced capture timer/Timer (ECT or TIM)	System Timer Module (STM)
Voltage Regulator (VREG)	RTC/API

A summary of the features for each MPC5604B peripheral are given and grouped in this way:

- CPU, interrupts, and XBAR

- Timers and ADC
- Clocks, power management, and Boot
- Communication
- GPIO
- Safety features

NOTE

Direct Memory Access (DMA) is available on MPC5602D and MPC5605/6/7B.

2.2.1 CPU, interrupts, and XBAR

2.2.1.1 e200z0h core

The e200z0h processors integrate an integer execution unit, branch control unit, instruction fetch and load/store units, and a multi-ported register file capable of sustaining three read and two write operations per clock. Most integer instructions execute in a single clock cycle. Branch target prefetching is performed by the branch unit to allow single-cycle branches in some cases. The e200z0h core is a single-issue, 32-bit PowerPC Book E VLE-only design with 32-bit general purpose registers (GPRs). All arithmetic instructions that execute in the core operate on data in the GPRs. Instead of the base PowerPC Book E instruction set support, the e200z0h core only implements the VLE (variable-length encoding) APU, providing improved code density. The VLE APU is further documented in the *PowerPC VLE APU Definition*, a separate document.

The following is a list of some of the key features of the e200z0h core:

- 32-bit single issue core, conforming to Power Architecture's Book E programmer's model
- Hard-wired to VLE (Variable Length Encoding) only mode, offering increased code density with only slight impact in performance over traditional Book E — 16-bit and 32-bit VLE instructions are supported
- Four-stage pipeline (instruction fetch, decode, execute and writeback)
- Pipeline that maps directly onto system bus so platform memories can be accessed with zero wait states
- Single-cycle execution for most arithmetic logical instructions
- Branches performed in one cycle (not taken), two cycles taken
- Load/store unit provides single-cycle loads and stores to and from GPRs
- Support of low-power modes with the ability to halt the core via wait instruction
- Fully synthesizable core
- Nexus class 1 debug built into core
- Big-endian only support

2.2.1.2 INTC — Interrupt controller

The interrupt controller in the platform takes all of the external peripheral interrupt requests and routes them to one of the sixteen core interrupts (IVOR4). The INTC supports the standard software vector mode interrupt handling as well as hardware vector mode which can reduce interrupt latency but with a code overhead. Further information is given in Section 5, "Interrupt table."

The INTC module has these features:

- Support of 140 peripheral and 8 software-configurable interrupt request sources
- Unique 9-bit vector per interrupt source
- Each interrupt source can be programmed to one of sixteen priorities
- Pre-emption
  - Pre-emptive prioritized interrupt requests to processor
  - ISR at a higher priority pre-empts ISRs or tasks at lower priorities
  - Automatic pushing or popping of pre-empted priority to or from a LIFO
  - Ability to modify the ISR or task priority — modifying the priority can be used to implement the priority ceiling protocol for accessing shared resources
- Low latency — three clocks from receipt of interrupt request from peripheral to interrupt request to processor

2.2.1.3 Crossbar (XBAR)

One of the most important parts of the platform architecture is the XBAR. Everything connected to the crossbar on the north side is a master and everything on the south side is a slave. The crossbar allows two simultaneous master/slave connections without any contention, in other words concurrent transactions can occur from any master port to any slave port. If the masters attempt to access the same slave port, then a pre-defined arbitration scheme takes effect, based on the priority that is assigned to the master.

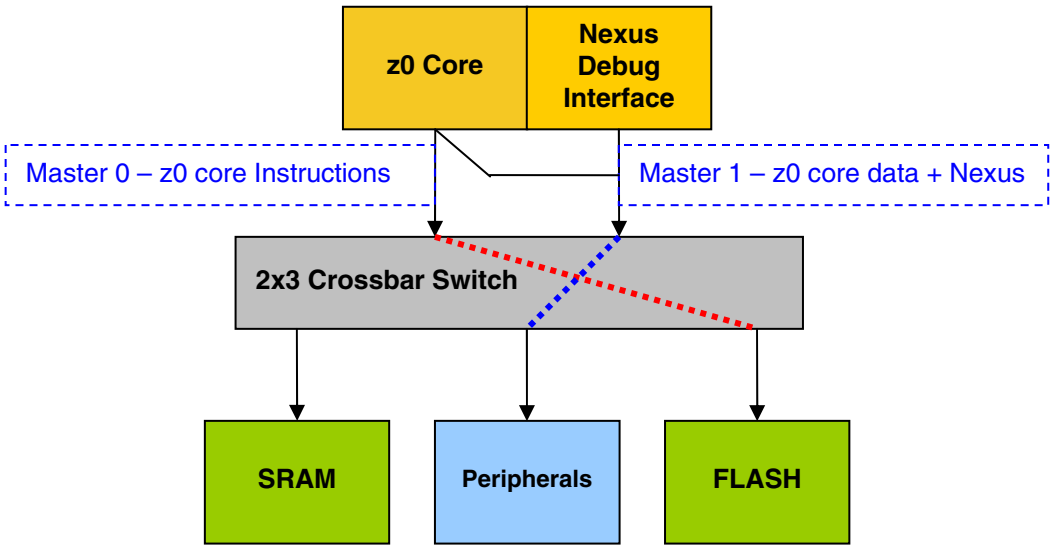

Figure 3. Crossbar switch, master, and slaves

The XBAR supports the following features:

- Two master ports:
  - Both master ports can access two different slave ports concurrently
  - Core: e200z0h core instructions
  - Core: e200z0h core data/Nexus
- Three slave ports
  - Flash (refer to the flash memory chapter for information on accessing flash memory)
  - Internal SRAM
  - Peripheral bridges
- Slave port contention (both masters trying to access the same slave) is arbitrated based on master priority
- 32-bit address and 32-bit data bus interface between two master ports and three slave ports
- No clock cycle delay across crossbar

2.2.2 Timers & ADC

The MPC5604B has a number of timers, two of which (eMIOS and PIT) can directly trigger conversions at the ADC without software interaction.

2.2.2.1 ADC

- 10-bit resolution
- Minimum conversion time of 1 µs
- 36 channels, expandable to 64 channels via external multiplexing
  - 16 high-precision channels
  - 20 extended channels, four being expandable to as many as 32 external channels
- External multiplier decoder signal generation
- Individual conversion registers for each channel (internal and external)
- Three different sampling and conversion time registers CTR[0:2] (internal precision channels, extended internal channels, external channels)
- 64 data registers for storing converted data (conversion information, such as mode of operation (normal, injected or CTU), is associated to data value)
- Conversion triggering sources:
  - Software
  - CTU
  - PIT channel 2 (for injected conversion)
- Four analog watchdogs
  - Interrupt capability
  - Allows continuous hardware monitoring of four analog input channels
- Presampling (V_{SS} and V_{DD})

- Conversions on external channels managed in same way as internal channels, ensuring transparency to the application
- One Shot/Scan mode
- Chain Injection mode to interrupt normal ongoing conversion
- Power-down mode
- 2 different abort functions allow ability to abort either single-channel conversion or chain conversion
- Auto-clock-off

2.2.2.2 eMIOS

The Enhanced Modular IO Subsystem (eMIOS) provides functionality to generate or measure time events.

- Two eMIOS blocks with 28 channels each
- Each eMIOS has one global prescaler
- Each channel has a 16-bit data register
- Each eMIOS has 5 × 16-bit wide counter buses used to drive other channels
  - Counter bus can further divide global clock by 1, 2, 3, or 4
- Motor control capability
- Channels can be configured to operate in the following modes:
  - General purpose input/output
  - Single Action Input Capture
  - Single Action Output Compare
  - Input Pulse Width Measurement
  - Input Period Measurement
  - Double Action Output Compare
  - Modulus Counter
  - Modulus Counter Buffered
  - Output Pulse Width and Frequency Modulation Buffered
  - Output Pulse Width Modulation Buffered
  - Output Pulse Width Modulation with Trigger

2.2.2.3 CTU — Cross Triggering Units

The Cross Triggering Unit (CTU) allows the synchronization of a single ADC conversion with a timer event from eMIOS or PIT.

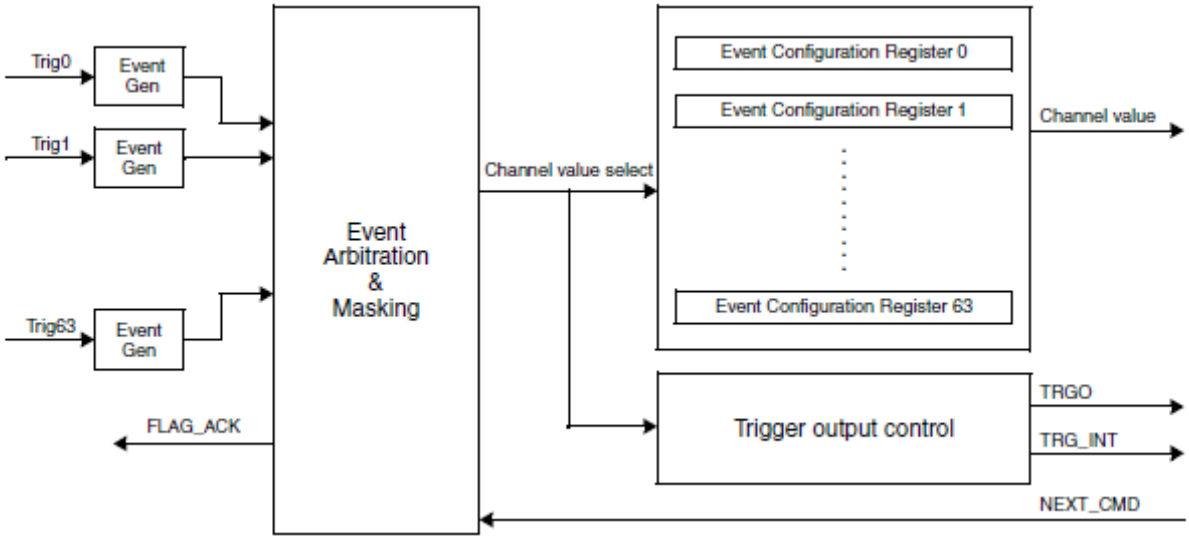

Figure 4. Block diagram of cross triggering unit

2.2.2.4 PIT — Periodic Interrupt Timer

The Periodic Interrupt Timer (PIT) is an array of six independent 32-bit timers that can be used to generate interrupts.

2.2.2.5 RTC/API — Real Time Counter/Autonomous Periodic interrupt

The RTC is a free running counter used for time-keeping applications. The RTC may be configured to generate an interrupt at a predefined interval independent of the mode of operation (run mode or low-power mode). If in a low-power mode when the RTC interval is reached, the RTC first generates a wakeup and then asserts the interrupt request. The RTC also supports an autonomous periodic interrupt (API) function used to generate a periodic wakeup request to exit a low-power mode or an interrupt request.

2.2.2.6 STM — System Timer Module

The System Timer Module (STM) is a 32-bit timer designed to support commonly required system and application software timing functions. The STM includes a 32-bit up counter and four 32-bit compare channels with a separate interrupt source for each channel. The counter is driven by the system clock divided by an 8-bit prescale value (1–256).

2.2.2.7 SWT – Software Watchdog Timer

Unlike other e200 cores, the e200z0h core has no embedded free-running counter. This means that there is no watchdog on the core. The Software Watchdog Timer (SWT) is a peripheral module that can prevent system lockup in situations such as software getting trapped in a loop or if a bus transaction fails to terminate.

The SWT has the following features:

- 32-bit time-out register to set the time-out period
- The unique SWT counter clock is the undivided slow internal RC oscillator 128 kHz (SIRC) — no other clock source can be selected
- Programmable selection of window mode or regular servicing
- Programmable selection of reset or interrupt on an initial timeout
- Master access protection
- Hard and soft configuration lock bits

2.2.3 Clocks, power management, and boot

2.2.3.1 ME — mode entry

The Mode Entry (ME) module allows the configuration and selection of a number of modes.

- Run modes:
  - DRUN (default run mode after reset)
  - RUN0, RUN1, RUN2, RUN3
- Low-power modes:
  - Stop
  - Halt
  - Standby
- Diagnostic and fault modes:
  - Test
  - Safe
  - Reset
- For each mode the following can be configured:
  - System clock: FIRC, FXOSC, or FMPLL
  - CFLASH: enabled, low-power, or power-down
  - DFLASH: enabled, low-power, or power-down
  - Pad control: enabled, disabled, or high impedance
  - Main voltage regulator: enabled or disabled

The mode entry allows each peripheral to be assigned to one of eight peripheral run mode subsets or one of eight peripheral low-power mode subsets. Each peripheral subset can then be enabled or disabled for each mode.

2.2.3.2 PCU — Power control unit

The power control unit is used to reduce the overall power consumption of the SoC. This is done by disconnecting parts of the SoC from the power supply. The power control unit (PCU) controls three power domains:

- PD0: Wake up from STANDBY circuit including API/RTC, WKUP, CANSAMPLER, RGM, PCU, ULPREG, FIRC, SIRC, and 8K RAM
- PD1: All other circuitry, excluding 24K RAM
- PD2: 24K RAM

The state of the power domains depends on the current mode:

- PD0: on for all modes
- PD1: on for all modes on except STANDBY
- PD2: user-defined, can be on or off for all modes

2.2.3.3 RGM — Reset Generation Module

The reset generation module (RGM) manages destructive and functional reset sources. The RGM can also control selection of alternate boot via the backup RAM on standby mode exit.

- Destructive reset sources:
  - Power-on reset
  - 1.2 V low-voltage detected (power domain #0)
  - 1.2 V low-voltage detected (power domain #1)
  - Software watchdog timer
  - 2.7 V low-voltage detected
- Functional reset sources:
  - External reset
  - JTAG-initiated reset
  - Core reset
  - Software reset
  - Checkstop reset
  - FMPLL fail
  - FXOSC frequency lower than reference
  - CMU clock frequency higher/lower than reference
  - 4.5 V low-voltage detected
  - Code or data flash fatal error

Some functional reset sources can be disabled so that either SAFE mode is entered or an RGM interrupt is generated.

Some functional reset sources can be shortened to reduce start-up time.

2.2.3.4 FMPLL — Frequency Modulated Phase Lock Loop

The FMPLL has the following features:

- Input clock frequency 4–16 MHz
- Voltage controlled oscillator (VCO) range 256–512 MHz

- Reduced frequency divider (RFD) for reduced frequency operation without forcing the FMPLL to re-lock frequency-modulated FMPLL
  - Modulation enabled/disabled through software
  - Triangle wave modulation
- Programmable modulation depth
  - ±0.25% to ±4% deviation from center spread frequency
  - 0.5% to +8% deviation from down spread frequency
  - Programmable modulation frequency dependent on reference frequency
- Self-clocked mode (SCM) operation
- Five available modes:
  - Normal
  - Progressive clock switching
  - Normal with SSCG
  - Power-down
  - 1:1 mode

2.2.3.5 CMU — Clock Monitor Unit

The Clock Monitor Unit (CMU) serves two purposes:

1. To supervise the integrity of the FXOSC and FMPLL. If the FMPLL leaves an upper or lower frequency boundary or the FXOSC fails, the CMU can detect and forward these events towards the ME and RGM. Under these conditions a number of options are available: reset the device, SAFE mode entry where the system clock defaults to FIRC, or generate an interrupt.
2. The CMU provides a frequency meter, which allows measurement of the frequency of one clock source versus a reference clock. This is useful for allowing the calibration of the FIRC as well as being able to calculate the time deviation of a counter which is clocked by the FIRC.

2.2.3.6 BAM — Boot Assist Module

The Boot Assist Module (BAM) is a block of read-only memory containing VLE code which is executed according to the boot mode of the device.

The BAM allows the ability to download code into internal SRAM through the following serial communication interfaces and execute it afterwards:

- FlexCAN
- LINFlex

2.2.3.7 ECSM

The Error Correction Status Module (ECSM) provides a myriad of miscellaneous control functions for the device, including:

- Program-visible information about configuration and revision levels

- A reset status register
- Wakeup control for exiting sleep modes
- Optional features such as information on memory errors reported by error-correcting codes

2.2.3.8 SSCM

The System Status and Configuration Module (SSCM) contains information about the current state and configuration of the system that may be useful for configuring application software and for debugging the system.

The SSCM includes these features:

- System configuration and status
  - Memory sizes/status
  - Device mode and security status
  - Determining the boot vector
  - Search code flash for bootable sector
- Device identification information (MCU ID registers)
- Debug status port enable and select
- Bus and peripheral abort enable/disable

2.2.3.9 WKUP — Wake-up Unit

The Wakeup Unit (WKUP) provides a means to wake the MCU from low-power modes and control non-maskable interrupt (NMI).

The WKUP supports the following features:

- Non-maskable interrupt support with:
  - One NMI source with bypassable glitch filter
  - Independent interrupt destination: non-maskable interrupt, critical interrupt, or machine check request
  - Edge detection
- External wakeup/interrupt support with:
  - Three system interrupt vectors for up to eighteen interrupt sources
  - Analog glitch filter for each wakeup line
  - Independent interrupt mask
  - Edge detection
  - Configurable system wakeup triggering from all interrupt sources
  - Configurable pullup
- On-chip wakeup support from API and RTC

2.2.4 Communication

2.2.4.1 DSPI

The Deserial Serial Peripheral Interface (DSPI) provides a synchronous serial bus for communication between the MCU and an external peripheral device.

The DSPI supports these SPI features:

- Full-duplex, three-wire synchronous transfers
- Master and slave mode
- Buffered transmit and receive operation using the Tx and Rx FIFOs, with depths of four entries
- Visibility into Tx and Rx FIFOs for ease of debugging
- FIFO bypass mode for low-latency updates to SPI queues
- Programmable transfer attributes on a per-frame basis
  - Six clock and transfer attribute registers
  - Serial clock with programmable polarity and phase
  - Programmable delays
    - CS to SCK delay
    - SCK to CS delay
    - Delay between frames
  - Programmable serial frame size of four to sixteen bits, expandable with software control
  - Continuously held chip select capability
- Up to six peripheral chip selects, expandable to sixty-four with external demultiplexer
- Deglitching support for up to thirty-two peripheral chip selects with external demultiplexer
- Two DMA conditions for SPI queues residing in RAM or flash:
  - Tx FIFO is not full (TFFF)
  - Rx FIFO is not empty (RFDF)
- Six interrupt conditions:
  - End of queue reached (EOQF)
  - Tx FIFO is not full (TFFF)
  - Transfer of current frame complete (TCF)
  - Rx FIFO is not empty (RFDF)
  - FIFO overrun (attempt to transmit with an empty Tx FIFO or serial frame received while Rx FIFO is full)
    - RFOF
    - TFUF
- Modified SPI transfer formats for communication with slower peripheral devices
- Supports all functional modes from QSPI sub-block of QSMCM (MPC500 family)
- Continuous serial communications clock (SCK)

2.2.4.2 FlexCAN

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification.

The FlexCAN module includes these features:

- Full implementation of the CAN protocol specification, version 2.0B
  - — Standard data and remote frames
  - — Extended data and remote frames
  - — 0–8 byte data length
  - — Programmable bit rate up to 1 Mbit/s
  - — Content-related addressing
- Flexible message buffers (up to sixty-four) of 0–8 byte data length
- Each message buffer configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask registers per message buffer
- Includes 1056 bytes (64 message buffers) of RAM used for message buffer storage
- Includes 256 bytes (64 message buffers) of RAM used for individual Rx Mask registers
- Full-featured Rx FIFO with storage capacity for six frames and internal pointer handling
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either eight extended, sixteen standard or thirty-two partial (eight bits) IDs, with individual masking capability
- Selectable backwards compatibility with previous FlexCAN version
- Programmable clock source to the CAN protocol interface, either bus clock or crystal oscillator
- Unused message buffer and Rx Mask register space can be used as general purpose RAM space
- Listen-only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Not dependent on transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages

2.2.4.3 CAN sampler

The CAN sampler peripheral is used to store the first identifier of CAN message “detected” on the CAN bus while no precise clock (crystal) is running. Typically the CANSAMPLER is used in low-power modes (STOP, HALT, or STANDBY) or in RUN mode with the crystal switched off.

2.2.4.4 I^2C — Inter-IC communication

The I^2C module has these features:

- Compatible with I^2C bus standard
- Multi-master operation
- Software programmable for one of 256 different serial clock frequencies
- Software-selectable acknowledge bit
- Interrupt driven byte-by-byte data transfer
- Arbitration lost interrupt with automatic mode switching from master to slave
- Calling address identification interrupt
- Start and stop signal generation/detection
- Repeated start signal generation
- Acknowledge bit generation/detection
- Bus busy detection

2.2.4.5 LINFlex

The Local Interconnect Network Flexible (LINFlex) controller interfaces the LIN network and supports LIN protocol versions 1.3; 2.0 and 2.1; and J2602 in both master and slave modes. The LINFlex also offers UART mode.

The LINFlex module has these LIN features:

- Master mode with autonomous message handling
- Classic and enhanced checksum calculation and check
- Single 8-byte buffer for transmission/reception
- Extended frame mode for in-application programming purposes
- Wakeup event on dominant bit detection
- True LIN field state machine
- Advanced LIN error detection
- Header, response, and frame timeout
- Slave mode
  - Autonomous header handling
  - Autonomous transmit/receive data handling
- LIN automatic resynchronization, allowing operation with 16 MHz fast internal RC oscillator as clock source
- 16 identifier filters for autonomous message handling in slave mode

The LINFlex module has the following UART features:

- Full duplex communication
- 8- or 9-bit with parity
- 4-byte buffer for reception, 4-byte buffer for transmission
- 8-bit counter for timeout management

Features common to both LIN mode and UART mode:

- Three operating modes for power saving and configuration registers lock:
  - Initialization
  - Normal
  - Sleep
- Two test modes:
  - Loop-back
  - Self-test
- Maskable interrupts
- Fractional baud rate generator

2.2.5 GPIO

This System Integration Unit Lite (SIUL) is used for the management of the pads and their configuration. It controls the multiplexing of the alternate functions used on all pads as well as being responsible for the management of the external interrupts to the device.

The System Integration Unit Lite supports these features:

- GPIO
  - GPIO function on up to 123 I/O pins
  - Dedicated input and output registers for each GPIO pin
- External interrupts
  - Two system interrupt vectors for up to 16 interrupt sources
  - Sixteen programmable digital glitch filters
  - Independent interrupt mask
  - Edge detection
- System configuration
  - Pad configuration control
    - Open-drain
    - Slew rate control
    - Pullup
    - Pulldown

2.2.6 Safety features

2.2.6.1 MPU — Memory Protection Unit

The Memory Protection Unit (MPU) provides hardware access control for all memory references generated in the device. Programmable region descriptors define memory spaces and their associated access rights. The MPU monitors all system bus transactions and evaluates the validity of each transfer. Memory references that have sufficient access control rights are allowed to complete, while references that

are not mapped to any region descriptor or have insufficient rights are terminated with a protection error response.

The MPU module provides these capabilities:

- Support for eight program-visible 128-bit (4-word) region descriptors
- Each region descriptor defines a modulo-32 byte space, aligned anywhere in memory
  - Region sizes can vary from a minimum of 32B to a maximum of 4GB
- Two types of access control permissions defined in single descriptor word
  - Processors have separate {read, write, execute} attributes for supervisor and user accesses
  - Non-processor masters have {read, write} attributes

2.2.6.2 Register Protection

The Register Protection module offers a mechanism to protect defined memory-mapped address locations in a module under protection from being written. The address locations that can be protected are module-specific.

The Register Protection includes the following features:

- Restrict write accesses for the module under protection to supervisor mode only
- Lock registers for first 6 KB of memory-mapped address space
- Address mirror automatically sets corresponding lock bit
- Lock bits can be protected from changes once they are configured

3 Startup flow

S12XE does not require any special boot sequences, which are necessary on MPC5604B.

The MPC5604B boot sequence from power-on reset is shown in Figure 5. There are two external boot configuration pins on MPC5604B (FAB and ABS) which determine whether the device boots into code located in flash or to serial boot mode, allowing code to be loaded and executed from the device RAM via either LIN or CAN.

When booting to flash, the SSCM (System Status and Configuration Module) searches the flash for a valid RCHW (Reset Configuration Half Word) from one of five possible locations. If a valid RCHW is found, execution is transferred to the flash address defined by the next 32-bit address immediately after the RCHW. If no valid RCHW is found, the SSCM places the device into static mode in order to conserve power.

When booting serially, execution is transferred to the BAM (Boot Assist Module) which reads the latched value of the ABS pin to determine whether LIN or CAN boot will be attempted. If a serial boot should fail, for example due to an incorrect serial password, then the MCU is placed into static mode after a pre-determined delay.

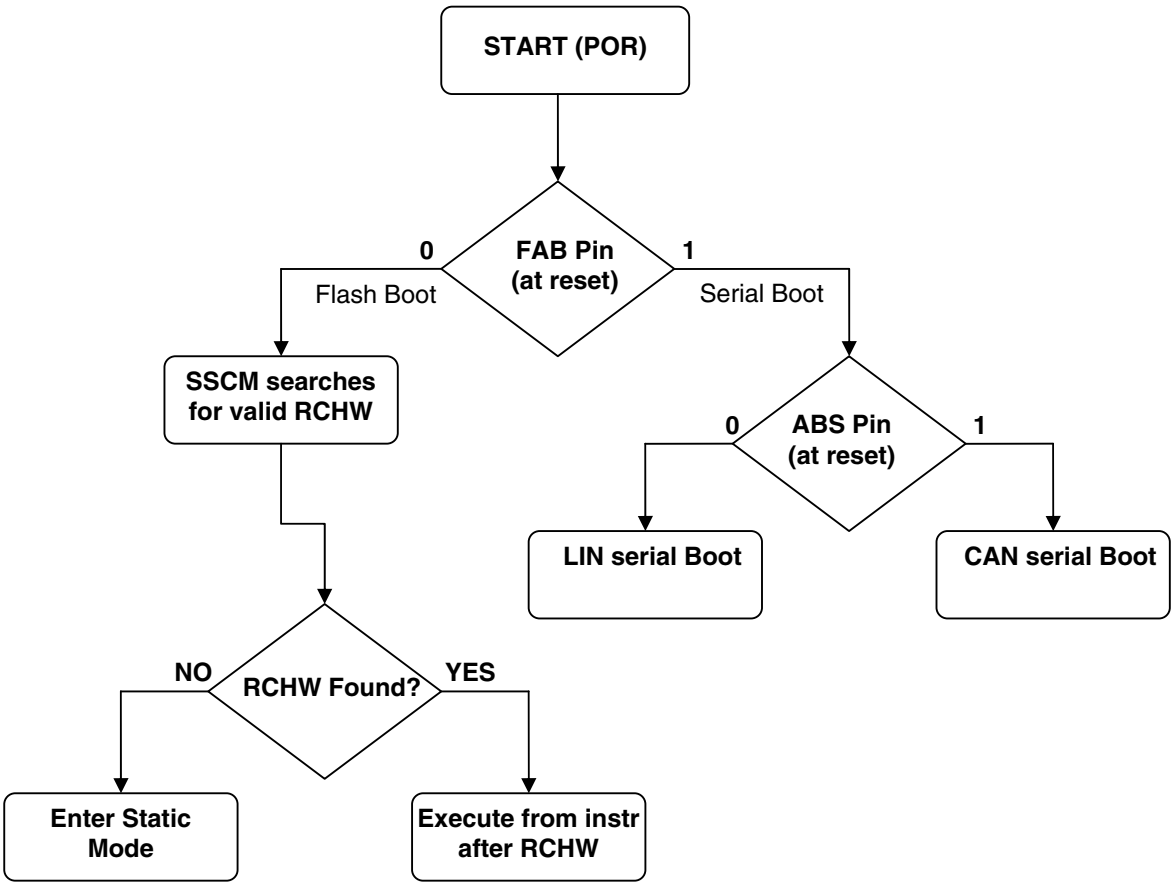

Figure 5. MPC5604B startup routine

4 Debug

The MPC5604B family uses a different development environment approach than the S12X but the features provided are similar. Many of the same vendors provide support for the two families and so the working environment in many cases appears identical. There are different standards for the connection of debuggers and for the debuggers themselves and so these are required changes. The tools available for the MPC5604B family are compatible with any MPC56xx and MPC55xx devices.

The debug facility is summarized in Table 3. However, although the interface is different, there are common features.

Table 3. Debugging information S12XE vs. MPC5604B

Function	S12XE	MPC5604B
Debug interface	Background debug mode (BDM)	Nexus2+[1]/JTAG
Trace capability	On-chip Debug Module (DBG)	External Nexus probe required

Table 3. Debugging information S12XE vs. MPC5604B (continued)

Function	S12XE	MPC5604B
Trace debug size	512-byte trace buffer	Tool-dependent
Low-power mode debug	No	Yes
Dedicated debug pins	One (BDM interface is six pins)	JTAG (6) / Nexus[1] (14)
Secure mode operation	Limited flash access/mass erase only	Flash access prevented

[1] Only available in 208 BGA package.

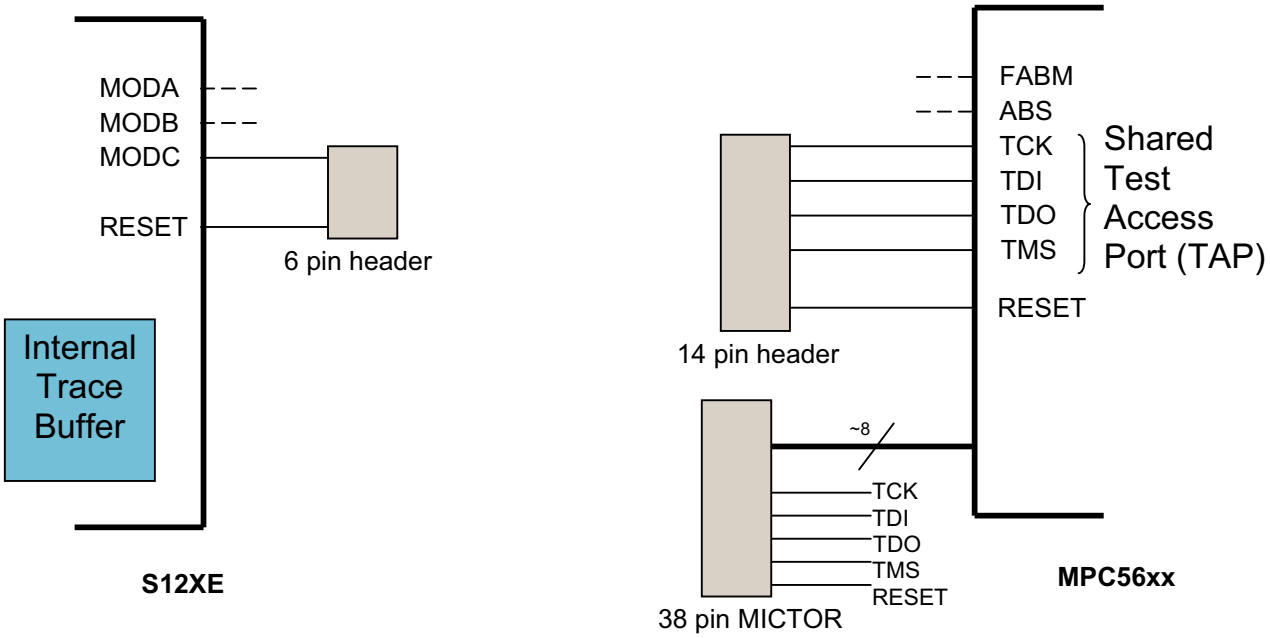

Figure 6. S12XE and MPC5604B debug pins

S12XE uses a single wire interface, the BKGD pin, to communicate both hardware and software commands in the debugging environment, which can be entered from reset or user code. MPC5604B contains the industry standard Nexus 2+ debug interface which is physically different from the BDM as it has 14-pin JTAG for static debug and 38-pin MICTOR connector for trace.

4.1 Debug trace

S12XE has four address and two data breakpoints, and performing a trace is done on the on-chip debug module (DBG). The debug module provides:

- On-chip trace buffer with flexible triggering
- Non-intrusive debugging of application software
- Debugging of both CPU12X and XGATE cores
- Start, mid-, and end-point triggering options based on four comparators
- Tracing disabled when device is secured
- Options for change of flow, to capturing everything

- 512-byte (64-line by 64-bit) trace buffer, accessible in a 16-bit window in RAM

After each complete 64-bit trace buffer line is read, an internal pointer to RAM is incremented so that the next read will receive fresh information. The trace information is retrieved via the BDM single wire connection from user software or from application software.

MPC5604B does not have an on-chip trace function. Instead it uses a debug engine which captures trace information and sends it to an external debugger. On the MPC5604B this is limited to a program trace where triggers can be set to start and stop the trace based on user configurable options. The resultant trace information is captured and analyzed by use of an external debug tool.

5 Interrupt table

The S12X device provides a single interrupt and reset table that contains all of the interrupt and reset vectors. The highest priority vectors are assigned to power-on or error condition resets. Next are the pseudo non-maskable (XIRQ) and external interrupt (IRQ) vectors, and finally the interrupts that belong to the on-chip peripherals — see Table 4.

Table 4. S12XE interrupt table

Vector	Interrupt/Reset
0xFFFE	POR
0xFFFE	LVR
0xFFFE	External
0xFFFE	Illegal address
0xFFFC	Clock monitor reset
0xFFFA	COP
0xF8	Unimplemented instruction
0xF6	SWI
0xF4	XIRQ
0xF2	IRQ
0xF0	RTI
0xEE	ECT0
0xEC	ECT1
0x14	MPU
0x12	SYS
0x10	Spurious Interrupt

Although this table appears to consist of a single group of vectors, the architecture actually implements the behavior as two separate tables. The upper part of Table 4 consists of the resets and interrupts that are part of the CPU architecture itself. The lower part of the table consists of the peripheral interrupt vectors that

share the CPU's IRQ function. These all require the IRQ to be enabled (CPU I-bit set) and if multiple peripherals cause an interrupt they have to be prioritized by the interrupt controller such that only one can trigger the IRQ. The S12X simplifies the system for users by automatically assigning the correct interrupt vector for the source of the IRQ, so fetching a peripheral interrupt is as seamless as fetching any of the true CPU hardware interrupts.

Table 5. True S12X interrupt architecture

Vector	Interrupt/Reset
0xFFFE	POR
0xFFFE	LVR
0xFFFE	External
0xFFFE	Illegal address
0xFFFC	CMU
0xFFFA	COP
0xF8	Unimplemented instruction
0xF6	SWI
0xF4	XIRQ
0xF2	IRQ

Interrupts handled by the interrupt controller via I bit

Automatic handling

0xF0	RTI
0xEE	ECT0
0xEC	ECT1
⋮	⋮
0x14	MPU
0x12	SYS
0x10	Spurious Interrupt

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

The MPC5604B family implements an interrupt mechanism that is very similar to that described above. There are dedicated hardware vectors that are used to detect CPU reset or error conditions and a general purpose "external interrupt" which provides the ability for peripherals to cause an interrupt. Like the S12X IRQ, this interrupt must be enabled before any peripheral interrupt can be taken, and the interrupt controller manages the priority when multiple requests occur. Also as with the S12X, there is a dedicated opcode to enable and disable this interrupt: wrteei 1 (or 0) on MPC5604B in place of CLI (or SEI) on the S12X.

The two main differences between the CPU interrupt tables are:

- On the MPC5604B the two interrupt tables appear more explicitly separate rather than integrated into a single list, and the interrupts from the two tables are handled in a slightly different way.
- The vector tables do not contain a vector that points to an interrupt handler. Instead the table allocates enough space to allow a branch to the vector address, and this branch is used to call the interrupt handler. The CPU interrupt table allocates sixteen bytes to each vector while the peripheral interrupt table allocates four bytes.

For the MPC5604B processors the CPU interrupt table is placed at an address in memory defined by the IVPR CPU register. The peripheral interrupt table is placed at a fixed offset of 0x0800 from the IVPR.

The MPC5604B CPU interrupt table contains several more entries than the S12X, as would be expected for a more complex processor. Each vector is given a number and labelled IVORn (for Interrupt VectOR n). Note that IVOR4 is the equivalent of the S12X IRQ function.

Table 6. Extracts of MPC5604B IVOR

IRQ #	Offset	Size (byte)	Interrupt	Module
Section A (Core Section)				
—	0x0000	16	Critical Input	Core
—	0x0010	16	Machine Check/NMI	Core
—	0x0020	16	Data Storage	Core
Section B (On Platform Peripherals)				
0	0x0800	4	Software Configurable flag 0	Software
1	0x0804	4	Software Configurable flag 1	Software
Section C				
38	0x0898	4	RTC	RTC/API
39	0x089C	4	API	RTC/API
Section D (Device Specific Vectors)				
157	0x0A74	4	EMIOS_GFR[F0,F1]	eMIOS1
158	0x0A78	4	EMIOS_GFR[F2,F3]	eMIOS1

The MPC5604B peripheral interrupt table behaves in a similar fashion, with each interrupt being allocated a given offset that allows a branch to the appropriate interrupt handler to be implemented.

5.1 Handling interrupts

As already described, the interrupt vector on the MPC5604B does not consist of a vector containing the address of the interrupt handler like the S12X. Instead it contains a branch instruction with the destination address being that of the interrupt handler.

Another major difference between the S12X and the MPC5604B is that the MPC5604B does not stack the CPU context when an interrupt occurs. This means that when handling an interrupt on the MPC5604B, it is usual to stack the CPU context in a small piece of code called the “prolog,” then call the interrupt service routine and finally restore the CPU context in an “epilog” routine before returning from the interrupt. In most cases a standard prolog and epilog can be used for all interrupt sources. Functional examples are provided by Freescale as part of the example application projects.

When handling CPU interrupts it may be necessary to clear certain CPU status flags, depending on the IVOR that occurred.

When handling peripheral interrupts there are further options and choices. All peripheral interrupts are passed to the interrupt controller that determines the priority of the interrupt and checks that against the

current interrupt priority level. If an interrupt is to be generated then this is passed to the CPU via the IVOR4 vector.

The Interrupt Controller provides a choice of two handling modes for peripheral interrupts:

- Hardware mode, in which the CPU automatically executes the handler code at the peripheral interrupt vector
- Software mode, in which the CPU automatically executes the handler code at the IVOR4 interrupt vector (in this case the interrupt controller IACKR contains the address of the peripheral interrupt vector and the handler should cause a jump to that address)

The two modes offer distinct performance and code size benefits. In hardware mode the interrupt handler begins to execute almost immediately after the interrupt is recognized. However, in this case each handler must contain a dedicated prolog and epilog that increases the program code size. In software mode the interrupt handler must first identify the source of the interrupt, causing a delay before the handler can begin to execute. However, in this case a single epilog and prolog can be shared among all interrupt vectors, thus saving program code size. The best choice will depend on the features of the application, including the number of active peripheral interrupts and the nature of the interrupt handlers themselves.

It is necessary to notify the interrupt controller when the handler has completed by writing to the EOIR register. The HVEN bit in the Interrupt Controller register controls hardware vector or software vector mode.

6 NVM memory

Although these devices have been manufactured from different technologies and operate in a different manner, similarities exist between the two. The S12X flash technology module contains program flash (P-flash), intended primarily for non-volatile code storage, and data flash (D-flash), which is used for non-volatile data storage or non-volatile storage to support emulated EEPROM, or a combination of both. The flash module on the MPC5604B (C90 LC 90 nm flash technology) also contains two flash arrays, known as code flash (C-flash) and data-flash (D-flash) which have the same functionality as P-flash and D-flash respectively on the S12X. The programmed and erased states for both technologies are the same: an erased bit reads one and a programmed bit reads zero.

The user should remember that the term ‘word’ refers to the number of bits that the CPU can handle at a time. So a word on S12X will be sixteen bits and on MPC5604B will be thirty-two bits. The S12X flash module may be read as bytes, aligned words (two bytes with even address), or misaligned words (two bytes with odd address). The read access time is one bus cycle for bytes and aligned words, and two bus cycles for misaligned words. P-flash can be programmed up to one phrase (an aligned group of four 16-bit words) in each P-flash block simultaneously, and D-flash can be programmed up to four words in a burst sequence.

The MPC5604B flash module is addressable by 32-bit word or double word (64 bits) for program, and page (128 bits) for read. Reads to the flash always return 128 bits, although read-page buffering may be done in the platform BIU. The address for each word retrieved within a page differs from the other addresses in the page only by address bits (3:2).

Both technologies rely on a controller to perform the flash operations. On S12X the memory map controller (MMC) is used to launch the various flash commands within a required sequence. The MMC

controls the paging capability to support the 8 MB global memory space and provide bus arbitration via 23-bit address load/store instructions. The user is required to update the flash common command object register (FCCOB) parameter fields and monitor FCCOB for results — the loading of the MMC is automatically completed.

On MPC5604B the platform flash controller is the interface between the crossbar switch and flash banks. It supports a 32-bit data bus at the AHB port and provides 128-bit read data interfaces to the C-flash and D-flash. The line buffers can be configured for instruction/data, so with careful planning this can mitigate against crossbar stalls because the core has separate masters for instruction and data fetch.

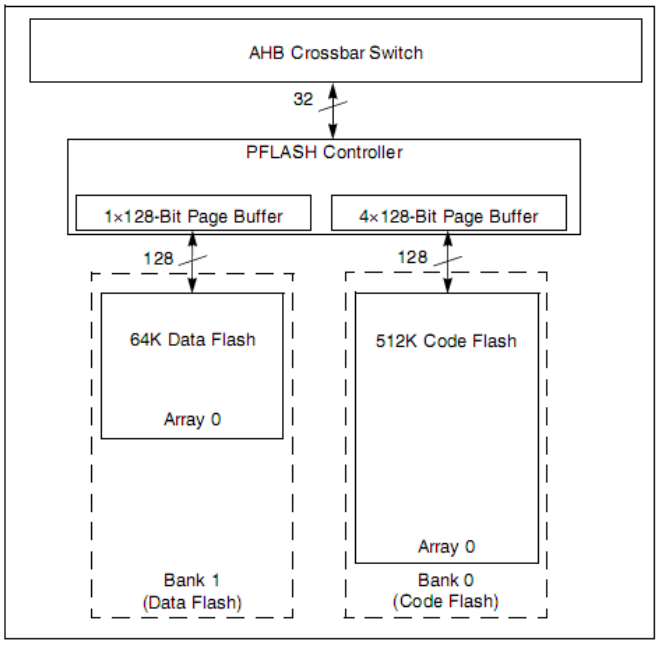

Figure 7. MPC5604B flash block diagram

Embedded within the memory interface is a hardware algorithm to program and erase the flash banks, known as the flash program/erase controller (FPEC). All the flash modify operations, such as program/erase/integrity check, are handled by the FPEC, through the flash user registers interface.

The sequences to perform a flash modify operation are via the following steps:

1. The first instruction selects the desired flash operation (equivalent to a flash command).
2. Define the operands, address, and data information.
3. Set EHV in MCR or AIE in UT0 to begin the flash operation.

It is not possible to modify the operands until the MCR.DONE (or UT0.AID) is set. These flash modify operations are completed when:

- MCR.DONE is set.
- Check MCR.PEG bit.
- Switch off FPEC by resetting MCR.EHV.
- De-select current flash operation by clearing MCR.PGM/ERS.

It should be noted that on both S12X and MPC5604B flash, it is not possible to access the same sector if a flash operation is occurring in this sector.

The following table summarizes and compares properties of each of the flash technologies:

Table 7. Comparison of S12X vs. MPC5604B flash

S12X	MPC5604B
MMC	Platform flash controller
P-Flash	C-Flash
D-Flash	D-Flash
ECC	ECC
Flash command operations	Flash modify operations/FPEC
Protection registers (FPROT/DFPROT)	Protection registers (PFAPR)
Programmed bit — logic low	Programmed bit — logic low
Erased bit — logic high	Erased bit — logic high

6.1 Error correction coding (ECC)

Both technologies are implemented with error correction codes (ECC) on their P-flash (C-flash) and D-flash memories which can resolve single bit faults and detect double bit faults. Program and erase protection are available on both memory technologies. On S12X the flash sectors for P-flash and D-flash are implemented via the FPROT and DFPROT registers. Within these registers it is possible to define regions of flash to protect, and access violation flags in the FSTAT register are used to identify whether any alteration of the protected regions has occurred. Similarly, on MPC5604B the platform flash controller provides programmable access protection for both read and write cycles from the master via the PFAPR register. Detection of a protection violation results in an error response from the platform flash controller on the AHB transfer.

6.2 Emulated EEPROM

Emulated EEPROM (EEE) is defined as a method to emulate the small sector size features and endurance characteristics associated with an EEPROM.

The primary difference between S12XE and MPC5604B with respect to EEPROM emulation is that on the S12XE family it is implemented in hardware, whereas on the MPC5604B it is implemented via software drivers (available from Freescale).

The S12XE family emulates EEPROM using up to a 4K buffer RAM and part, or all, of the D-flash. The buffer RAM acts as the volatile memory space for EEEPROM, where the D-flash space is used as the non-volatile memory space. The user must take care to configure the flash module memory controller and use the provided commands to configure module resources for emulated EEPROM operation.

On the MPC5604B, software drivers are used to control a portion of the on-chip D-flash which can optionally be assigned to EEEPROM. These software drivers are provided free by Freescale. See Section 9, “Reference material,” for useful links.

The MPC5604B flash is partitioned into 16 KB blocks. Two or four of these 16 KB blocks can be used for emulated EEPROM. By evenly distributing write/erase cycles across the flash blocks, write/erase cycle counts of more than one million can be achieved. The longer the data record size, the more efficient the scheme can be. Each flash block employed, and each data record, has a status field to ensure data integrity. Blocks are swapped from active to inactive in order to improve data record distribution and therefore write/erase cycle performance.

Data record read time will be dominated by the length of the data record and software search time. Optionally however, some system RAM could be used as cache to speed up the search process. Data record write time will be dominated by flash program time. The free EEE software driver from www.freescale.com contains an example of emulating EEPROM on the MCP5604B.

Table 8. S12XE vs. MPC5604 EEE

S12XE	MPC5604B
16-bit word size	32-bit word size
Hardware implementation	Software driver implementation
FTM (D-flash)	Data flash
Dedicated buffer RAM to mirror Flash — up to 4K	Uses standard Flash blocks — two or four 16 KB flash blocks with multiple read-while-write partitions.
Word (16-bit) Aligned (optimal) Byte aligned supported Misaligned word supported	Record scheme allows storage of variable size EE data elements in flash
Hardware ECC for EEEPROM	Hardware ECC for EEEPROM

7 Low-power considerations

In many embedded devices, the majority of the time is spent in a low-power/monitoring state. Both microcontroller families have been designed with this feature and have several operating modes to facilitate it.

S12XE has three low-power operating modes: wait, pseudo-stop, and stop. Stop mode is the lowest power configuration and wait mode gives the minimal power saving for the device by halting the CPU only. Table 9 summarizes the S12XE low-power modes.

Table 9. S12XEQ512 low-power modes

	External oscillator	System clock	PLL (FMPLL)	CPU activity	Optionally available modules/functions	Typical I_{DD}
Run	On	On	On/Off	On	All	72 mA
Wait	On	On	On/Off	Off	All	50 mA
Pseudo-stop	On	Off	Off	Off	RTI, COP, API, and ATD	185µA
Stop	Off	Off	Off	Off	API, ATD	30µA

MPC5604B has five highly configurable run modes — DRUN & RUN[0..3] — whereby the C-flash and D-flash, external oscillator, and PLL can be enabled or disabled. The three low-power modes — halt, stop,

and standby — are also highly configurable and offer low-power I_{DD} figures comparable to S12XE. In addition to the many mode options, the core offers the 'wait' instruction which can be used to temporarily stop the core clock for immediate reduction in run current. For wait, the core clock is restored with any interrupt source.

Table 10. MPC5604B low-power modes

	Internal RC oscillator (FIRC)	External oscillator (FXOSC)	PLL (FMPLL)	CPU clock	Peripheral clock	Optionally available modules/functions	Typical I_{DD}
DRun, Run	On	On/off	On/off	On	On/off	All	60 mA
Halt	On/off	On/off	On/off	Off	On/off	All	8 mA
Stop	On/off	On/off	Off	Off	On/off	All	180 μA
Standby	On/off	Off	Off	Off	Off	WKUP pins, API/RTC, CAN Sampler	20 μA

7.1 Low-power mode entry and exit

The S12X can be thought of as having a partially integrated operating mode architecture. This means that low-power modes are controlled centrally via the CPU instruction set using the WAI and STOP opcodes but many peripherals must be individually configured to be active or inactive in the modes. A reset or any interrupt, including pin interrupts (IRQ/XIRQ), can wake the S12XE device from low-power mode.

The MPC5604B has a much more sophisticated approach to managing operating mode and power management that centralizes configuration and control into a tightly linked group of modules. In this implementation, multiple operating modes are configured using the Mode Entry (ME), Clock Generation (CGM), Power Control (PCU), and Reset Generation (RGM) modules, and a single point of control allows the user to completely reconfigure the active portions of the MCU with a single command. Unlike the S12X the default condition of the MPC5604B out of reset requires active configuration of the control modules before peripherals are available for use.

Prior to entering one of the MPC5604B low-power modes — halt, stop, or standby — the user must configure a wakeup source such as an ISR (peripheral interrupt service request) or WKUP (hardware wakeup trigger) request to ensure the device can recover to a run mode. For clarity, note that 'wakeup' is the generic term for exit from low-power mode, whereas WKUP is a specific hardware wakeup trigger. Furthermore, before entering the low-power mode the mode must be configured:

1. All clocks (FXOSC, FIRC, SXOSC, SIRC, and FMPLL), C-flash, D-flash, RAM, main voltage regulator, and GPIO have to be configured for the specific low-power mode.
2. Each peripheral has to be assigned to a peripheral subset. The clock for each peripheral subset can be enabled or disabled for the specific low-power mode.

Available configuration options for each low-power mode is summarized in Table 11.

Table 11. MPC5604B low-power mode resource control

Resource	Control register	Halt	Stop	Standby
FIRC	ME_<mode>	On/off	On/off	On/Off
FXOSC	ME_<mode>	On/off	On/off	Off
SIRC	CGM_SIRC	On	On	On/Off
SXOSC	CGM_SXRC	On/off	On/off	On/Off
Peripheral Clocks	ME_PCTL[0..104] ME_LPPC[0..7]	On/off	On/off	Off
FMPLL	ME_<mode>	On/off	Off	Off
CFLASH	ME_<mode>	On/low power	On/power down	Power down
DFLASH	ME_<mode>	On/low power	On/power down	Power down
MVREG	ME_<mode>	On/off	On/off	Off
GPIO	ME_<mode>	On	On/	Tri-state
RAM	PCU_PCONF[2]	24 KB/48 KB	24 KB/48 KB	8 KB/3 2KB

The MPC5604B offers eighteen WKUP pins that can be triggered on rising and/or falling edge and each pin has the option of glitch filter and internal pullup. The API and RTC timers offer two further WKUP sources. Any WKUP event can generate an ISR as the twenty WKUP sources are split between three interrupt vectors.

For low-power mode exit there are subtle differences between halt, stop, and standby.

- The only way to exit halt mode and return to Runx is via an ISR. A WKUP source cannot wake from halt mode unless its corresponding interrupt is enabled.
- From stop mode a WKUP or an ISR (when system clock is enabled) can restore the MCU to Runx mode. The WKUP trigger will restore the Program Counter (PC) to the instruction address that the core was executing prior to stop mode entry. It should be noted that in stop mode, if the system clock is disabled, then the only way to exit is via a WKUP trigger. For this case, once the PC is restored and the core is enabled any pending interrupt will be serviced.
- Important caveats regarding interrupts and halt and stop low-power modes:
  - For both halt and stop mode any pending interrupt will prevent the device from entering the low-power mode, even for the stop mode case where the system clock is disabled and the same interrupt would not cause a wakeup once the device entered stop mode.
  - A wakeup from halt or stop on an interrupt will occur regardless of the configuration of the interrupt at the INTC.
- While in standby mode only a WKUP source can return the MCU to DRun mode. In standby mode the only circuitry remaining powered is that required to wake the MCU, such as the WKUP, API/RTC, CAN Sampler, RGM, PCU, FIRC (optional), SIRC (optional), 8 KB RAM, 32 KB RAM (optional) and low-power voltage regulator (LPVREG). With this always-on power domain, all system and peripheral configurations have to be restored. Once the INTC is configured and enabled, any pending WKUP interrupt will be serviced.

Table 12. MPC5604B available wakeup sources for low-power modes

MPC5604B mode	Wakeup source
Halt	ISR[1], ISR-WKUP[2]
Stop — system clock enabled	WKUP[3], ISR, ISR-WKUP
Stop — system clock disabled	WKUP
Standby	WKUP

1 ISR — Interrupt service request from any peripheral excluding WKUP.
2 ISR-WKUP — Interrupt service request from wakeup source.
3 WKUP — Wakeup source including WKUP pins and API and RTC.

The program flow at the exit from low-power mode for MPC5604B depends on the specific low-power mode and the wake-up trigger. The table below summarizes the program flow for all low-power mode exits and their wake-up trigger.

Table 13. MPC5604B wakeup from low-power mode program flow

MPC5604B Mode	ISR	WKUP	ISR-WKUP	Wakeup trigger	Program flow after low-power mode exit
Halt	—	—	—	ISR	ISR
	—	—	Enabled	WKUP	ISR-WKUP
Stop — system clock enabled	Enabled	—	—	ISR	ISR
	—	Enabled	Disabled	WKUP	PC
	—	Enabled	Enabled	WKUP	PC -> ISR-WKUP
Stop — system clock disabled	—	Enabled	Disabled	WKUP	PC
	—	Enabled	Enabled	WKUP	PC -> ISR-WKUP
Standby	—	Enabled	Disabled	WKUP	Valid RCHW in FLASH or 0x4000_0000 (RAM)
	—	Enabled	Enabled	WKUP	Valid RCHW in FLASH or 0x4000_0000 (RAM) ISR-WKUP will only be executed after interrupts have been configured at INTC

NOTE

The WKPU pins are always powered regardless of whether or not they are enabled at WKUP_WRER. Therefore all WKUP pins have to be properly terminated with either an external pullup, external pulldown, or the internal pullup at WKUP_WIPUR.

8 Hardware considerations

This section discusses the optimum method to lay out the PCB which will contain the device and other important components, as well as making reservations for the power supplies and grounding requirements. The application and environment tend to have the greatest impact on the latter parameters and the hardware should be built around these. Suffice to say, S12X and MPC5604B both follow the same rules given below, except for where their unique properties require additional considerations.

8.1 Power supply and PCB layout

Each device can operate within a range of 3.3–5 V, which is supplied externally to the MCU and is used to power the internal voltage regulators. Similarities exist whereby the external supply (3.3–5 V) is used for I/O buffers ADC and VREG, whereas the voltage regulator provides a power supply for the core, PLL, and RAM/NVM. Both families share a common feature: the core logic operating voltage is lower than the typical application operating voltage. For S12XE, the core logic operates at 1.84 V, and for the MPC5604B family the core logic operates at 1.2 V. In both cases the devices contain voltage regulators that take the 3.3 V or 5 V application voltage and provide the core with the appropriate supply. Like the S12X, the MPC5604B features a number of low-power modes (for example stop mode) that make use of progressively more efficient voltage regulators to reduce power consumption.

S12X contains a single voltage regulator which can be operated in various modes, and is capable of providing three separate supplies consisting of: 2 × 1.84 V and one 2.82 V from a 3.3/5 V external input (VDDR). No external components are necessary in line with the S12X VREG.

The MPC5604B VREG consists of three regulators for high, low, and ultra-low power regulation. An external capacitance connected close to the associated pins is required to provide a stable low voltage digital supply for the device. The output of the VREG is a 1.2 V supply to the internal device logic.

It should be noted that the MPC5604B device is partitioned into two separate power domains, PD0 and PD1. The voltage regulator is connected to each power domain by power switches which are programmable by software (through the PCU_PCONFn registers) and it is possible to reduce the power consumption by disconnecting a power domain. The maximum power saving is achieved in STANDBY0 mode.

Table 15 identifies the pins around the MCU which have a similar function.

Table 14. S12X/MPC5604B VDDx and VSSx pin comparison

	S12X		MPC5604B	
Port pin	**Pin Name**	**Typical Supply (V)**	**Pin Name**	**Typical Supply (V)**
Digital supply voltage	VDDx	3.13–5.50	VDD_HV	3.13–5.50
Digital supply ground	VSSX	0.00	VSS_HV	0.00
Voltage regulator	VDDR	3.13–5.50	VDD_BV	3.13–5.50
NVM power supply	VDDF	2.82	VDD_LV	1.20
Analog power supply	VDDA	3.13–5.50	VDD_HV_ADC	3.13–5.50

Table 14. S12X/MPC5604B VDDx and VSSx pin comparison (continued)

	S12X		MPC5604B	
Port pin	**Pin Name**	**Typical Supply (V)**	**Pin Name**	**Typical Supply (V)**
Analog low reference	VRL	0.00	VSS_HV_ADC	0.00
Analog high reference	VRH	3.13–5.50	VDD_HV_ADC	3.13–5.50
Oscillator/PLL supply	VDDPLL	1.84	VDD_LV	1.20
Oscillator/PLL ground	VSSPLL	0.00	VSS_LV	0.00

S12XE is a 0.18 µm flash technology whereas MPC5604B is 90 nm flash, hence the difference for the flash supply voltage.

For both devices:

- Every supply pair must be decoupled by a capacitor connected as near as possible to the corresponding pins.
- Use low ohmic low inductance connections between ground pins of the supply pairs.

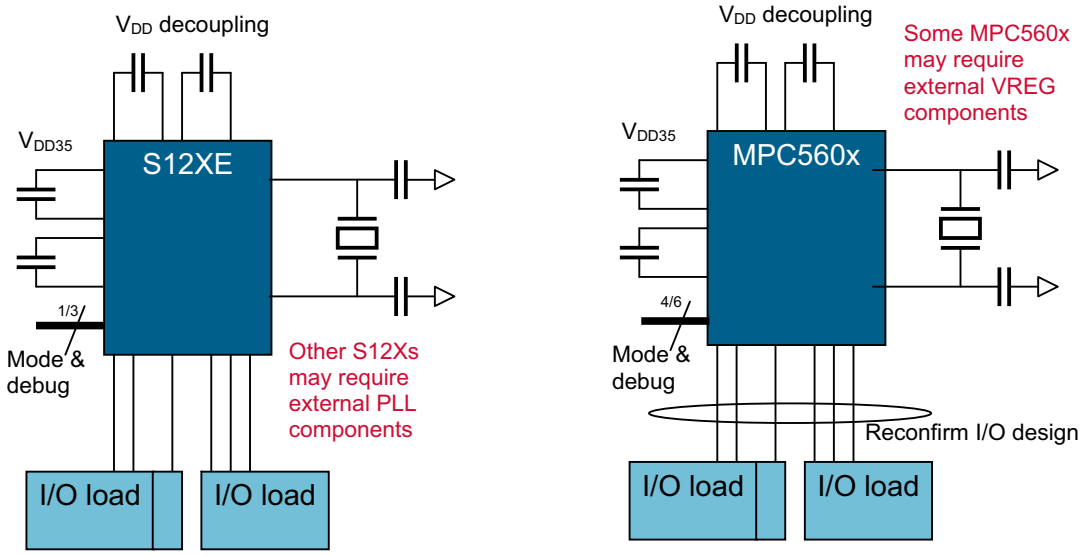

Figure 8. Peripheral connections to the MCU

There are no real differences between the devices when it comes to grounding, but the MPC5604B will have more power supplies.

8.2 Clocks and oscillators

Both devices use a similar clock structure, in that a Pierce configuration is common and both support external crystal/resonators in the range 4–16 MHz which can be operated with the PLL. Moreover, FMPLL and clock monitor functions are available on both devices.

Differences emerge in that MPC5604B supports a series of clock sources, whereas S12XE does not.

- External fast crystal oscillator 4–16 MHz (FXOSC)
- Internal fast IRC capable of 16 MHz operation (FIRC)
- Two low power oscillators:
  - Slow internal 128 kHz RC (SIRC)
  - Slow external 32 kHz crystal (SXOSC)

The clock generation module selects one of the clock sources as described above and contains the registers responsible for dividing the clock source. Unlike S12X, the user must configure the mode entry module, which is responsible for controlling the system clock selection. However, this module contains a clock switch which will enable the system clock and any clock dividers programmed via the clock generation module. The MPC5604B will start up with FIRC (without the PLL) unless software has been programmed to dictate otherwise.

Hardware design rules for oscillator layout are the same for both S12X and MP5604B — there are no special/different conditions which have to be applied. If S12XD was previously being used, then loop filter PLL components on the XFC pin will no longer be required. Common design rules listed below should be adhered to.

- Tank capacitors and feedback resistors are best selected from crystal/resonator manufacturer recommendations. It is also desirable for the crystal/resonator manufacturer to validate the application board for the best oscillator and/or components.
- Keep traces of oscillator pins, EXTAL, and XTAL as short as possible, and occupied board area for oscillator tank capacitors and crystal/resonator as small as possible.
- Do not place other signals or supplies underneath area occupied by oscillator tank capacitors and crystal/resonator, as well as the connection area to the MCU.
- Use suitable grounding — some designs require separate oscillator grounding.

8.3 Port considerations

The port integration module (PIM) is responsible for the management of the I/O port on S12XE. The equivalent module on MPC5604B is the system integration unit (SIU). The SIU controls the multiplexing of the alternate functions (GPIO, pad configuration, and interrupts) used on all pads as well as being responsible for the management of the external interrupts to the device. The S12XE functionality, such as input/output, pullup/pulldown control, slew rate, and open-drain control, is available on MPC5604B.

There is not much difference from an S12X application in terms of connecting these pins to external peripherals. The input current and voltage, input leakage, and pullup/pulldown characteristics are similar, whereas the output high/low voltage figures are slightly different. Table 15 compares some of the parameters.

Table 15. Voltage high and low level and leakage current of S12XE vs. MPC5604B

Parameter	Conditions	Value	
		S12XE	MPC5604B
Input high voltage (V)		$0.65 \times V_{DD35}$	$0.65 \times V_{DD35}$
Input low voltage (V)		V_{SS35}–0.3	V_{SS35}–0.3
Output high voltage (V)		V_{DD}–0.4	$0.8\ V_{DD}$
Output low voltage (V)		0.4	$0.1\ V_{DD}$
Input leakage current (nA)	Ambient temp	1	2

8.3.1 Functional pins

The following table lists control pins which are available on the MPC5604B device that require special consideration at the design stage. Where applicable, the equivalent S12X pin is listed for comparison.

Table 16. S12XE vs. MPC5604B control pins

Pin		Function
S12X	MPC5604B	
Test	—	For internal use only and should always be grounded.
Reset	Reset	Open drain/active low bi-directional pin — on reading high MCU test the internal reset sources and take the appropriate reset vector.
XTAL	EXTAL[1]	Analog output of oscillator amplifier circuit.
EXTAL	XTAL[2]	Analog input of oscillator amplifier circuit.
ECLK	CLKOUT	Outputs the frequency of the system clock.
—	FAB	Force Alternative Boot Mode — the pin is asserted prior to reset. State is used to determine whether serial boot from LINFlex or FlexCAN.
—	ABS	Alternative Boot Selector — Selects type of alternative boot — LINFlex or FlexCAN.

1 Analog output of the oscillator amplifier circuit, when the oscillator is not in bypass mode. Analog input for the clock generator when the oscillator is in bypass mode.

2 Analog input of the oscillator amplifier circuit. Needs to be grounded if oscillator is used in bypass mode.

On MPC5604B it should be noted that the oscillator pins EXTAL and XTAL can be used as either input/output or output/input depending on whether bypass mode is enabled or not. When the oscillator is not in bypass mode, EXTAL is output and XTAL is input. When the oscillator is in bypass mode, EXTAL is input — therefore the hardware design should consider this. Furthermore, the device will have two sets of oscillator pins, one for use with the slow external 32 kHz crystal (SXOSC) and the other for fast external crystal (FXOSC). The SXOSC pins are multiplexed with GPIO, which will be user configurable.

8.3.2 Debug pins

S12X has a single debug pin MODC/BKGD, whereas MPC5604B does not use BDM for debug purpose; this is done via four pins for static debug and fourteen pins for full Nexus trace. The pins of interest to the user are test data input (TDI), test data output (TDO), test mode select (TMS), and test clock input (TCK). These pins are shared through the Nexus development interface through the test access port (TAP) interface further described in the debug section.

9 Reference material

9.1 Application notes and software links

- AN2865 – MPC55xx/56xx Software Examples
- AN4036 - Migration within the MPC560xB/C/D Family
- AN3753 - MPC551x to MPC560xB/C, Migration
- AN3836 - Advanced Headlights Control and Diagnostics
- MPC56xx Flash software driver
- MPC56xx Emulated EEPROM driver
- MPC5604B Header files
- e200z0 Core Reference Manual

9.2 Hardware and development boards

- Evaluation board
- Motherboard and mini-modules
- EVB Users Guide

9.3 Part numbers

Table 17. Device identification

Device manufacturer	Part numbers	Maskset		JTAG ID	
		Cut1.1	Cut2	Cut1.1	Cut2
Freescale	MPC5604B, MPC5604C	2M07N	0M27V	0x1AE4001D	0x0AE4101D

9.4 MPC56xx family nomenclature

MW: 496 CSP

10 Revision history

Table 18. Changes made April 2012[1]

Section	Description
Front page	Add SafeAssure branding.
Title and section 1	Add Qorivva branding.
Back page	Apply new back page format.

1 No substantive changes were made to the content of this document; therefore the revision number was not incremented.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN4143
Rev. 1
06/2010

freescale™