Document Number:
Rev. 1, 01/2010

eMSG Inbound Context Management

by: Freescale Semiconductor, Inc.

eMSG is a completely new RapidIO Rev 2.1 compliant endpoint implementation for DSP products. The eMSG unit supports previously available Type 10 doorbells and Type 11 messaging and adds Type 9 data streaming.

eMSG also implements extensive Quality-of-Service features that support multiple prioritized streams of traffic between endpoints.

With the addition of Type 9 data streaming, management of inbound hardware reassembly context resources becomes an important consideration as system designers map their application to RapidIO-based systems. This White Paper will discuss their basic functionality and basic rules for how they should be managed.

Contents

1 Background

The RapidIO specification defines three forms of address-less messaging transactions: Type 9 data streaming, Type 10 doorbells and Type 11 data messages. Type 9 and Type 11 transactions carry data. A Type 10 doorbell carries a 16-bit info field but no data payload and is always a single RapidIO packet.

When a packet carries data, the maximum payload size is 256 bytes. Type 9 and Type 11 RapidIO transaction types define a maximum PDU of 64KB and 4KB respectively.   To carry PDUs larger than the native packet payload size, the protocol supports hardware segmentation and reassembly of the message payload.

The RapidIO specification requires that the receiver acknowledge back to the transmitter each Type 10 doorbell message and Type 11 message segment sent. When the receiver cannot temporarily accept the packet, the endpoint may send a retry to the sender causing the segment to be resent again. If an error occurs, it may send an error response. This logical-layer acknowledge (ACK) is distinct from the link-level acknowledge that occurs as the transaction crosses each link in the network. Logical retries can cause Type 11 message segments to arrive out of order and the protocol and hardware must be capable of reordering the segments in memory.

2 Segmentation and Reassembly

Segmenting outbound messages is relatively straightforward in hardware. However, reception of inbound segments and their reassembly is significantly more complex.

There are three steps to this process:

- Packet classification
- Reassembly resource management
- Writing payload and descriptor to memory

Figure 1 illustrates this process in more detail. The inbound packet must first be classified against expected inbound streams. After classification, it is identified as either the first segment of a new message or a middle/last segment of an existing one. If the packet is the first segment of a new message, hardware resources are allocated to track and reassemble the new message. When the last segment of a message arrives, these resources are released. Once each segment is associated with a reassembly resource, the segment and then the descriptor are written to memory. Processing concludes when the descriptor is issued to memory.

To maximize system performance, inbound messaging hardware should allow messages from one or more sources to be received at the same time. This implies that individual segments from these messages are interleaved in arbitrary ways. RapidIO packet headers allow hardware to distinguish a segment of one message from another and reassemble segments into their respective messages.

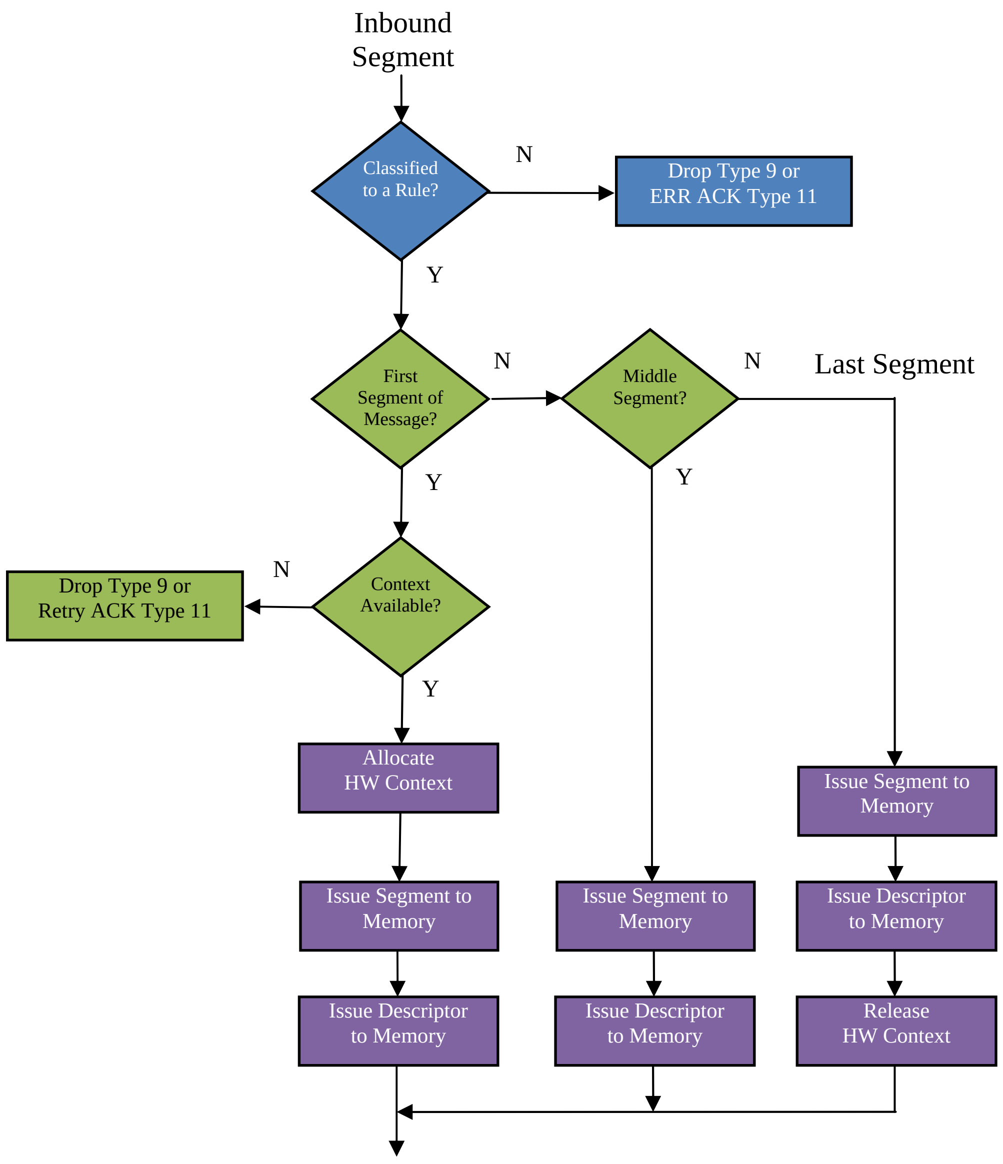

Figure 1 Inbound Segment Processing

3 Hardware Reassembly Contexts

As a message is transmitted segment by segment, a corresponding hardware reassembly resource (or context) for the message is referenced at the destination. Due to the complexity of this hardware resource, the number of contexts that can be implemented is usually limited. The number supported defines the maximum number of simultaneous outstanding messages that may be in-flight to a given destination endpoint. Software can determine the number of contexts supported by an endpoint by accessing the standardized Data Streaming Information CAR register (DSICAR). eMSG for the MSC8155 implements 24 reassembly contexts (or HW contexts) usable by Type 9-11 transaction types.

As shown in Figure 1, if the first segment of a new message arrives and is properly classified but all available HW contexts are holding other active messages, the behavior of the inbound eMSG hardware depends on the transaction type. Type 10 or 11 transactions cause hardware to issue a logical retry in the expectation resources will be available soon. Type 9 transactions are dropped because there is no mechanism for communicating an error or retry back to the receiver.

HW contexts within eMSG may be taken by Type 9, 10 or 11 messages. To avoid Type 11 logical retries or Type 9 packet loss, system designers must ensure at each endpoint that the maximum number of simultaneous inbound interleaved messages of all types never exceeds the available HW contexts.

As shown in Figure 2, each HW context entry can have three states:

- Invalid: Empty entry
- Open: Not all message segments have been received
  - State valid only for a multi-segment Type 9 or 11 message
  - Waiting for rest of segments to arrive
- Closed: All segments received
  - State valid for a single-segment Type 9-11 message or a multi-segment Type 9 or 11 message after all segments of the message have been received
  - Message is now guaranteed to complete
  - Remain in this state until associated command descriptor update is issued to memory

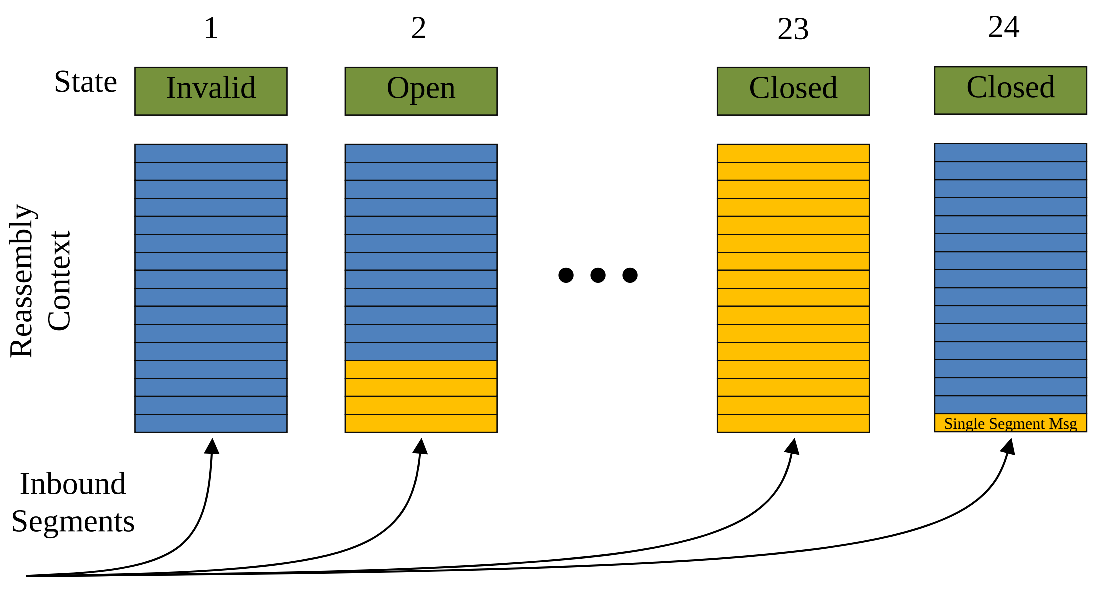

Figure 2 Hardware Reassembly Contexts

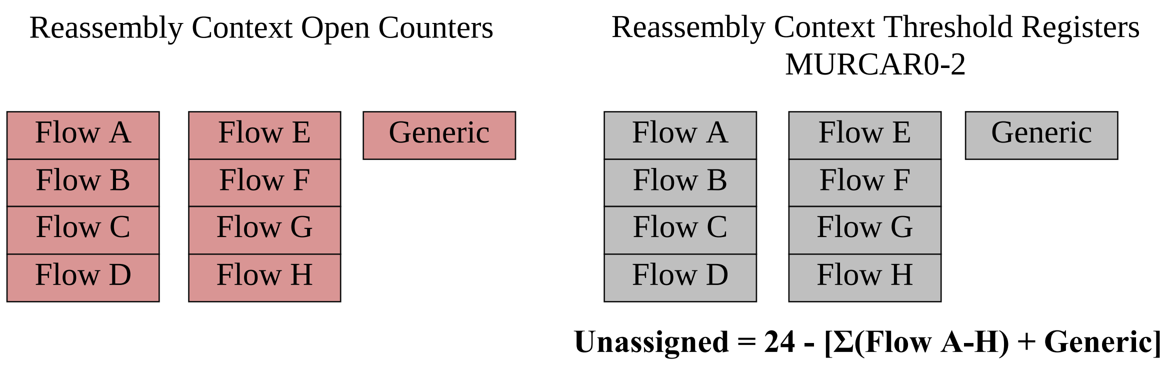
Reassembly Context Open Counters
Flow A
Flow B
Flow C
Flow D
Flow E
Flow F
Flow G
Flow H
Generic
Reassembly Context Threshold Registers MURCAR0-2
Flow A
Flow B
Flow C
Flow D
Flow E
Flow F
Flow G
Flow H
Generic
Unassigned = 24 - [Σ(Flow A-H) + Generic]

Figure 3 Per-Flow HW Context Counters and Threshold Registers

When a Type 10 or single-segment Type 9 or 11 message arrives and is classified, an invalid HW context is allocated and its state is moved from Invalid to Closed.

When the first segment of a new Type 9 or 11 multi-segment message arrives, an invalid HW context is allocated and its state is moved to Open. The entry is moved to Closed when the last segment has been classified to this context. Since single-segment messages move the HW context state directly from Invalid to Closed, the number of contexts in the Open state always equals the total number of outstanding multi-segment messages.

After the last segment arrives, processing concludes by writing the command descriptor to memory. At this point, the HW context state transitions from Closed to Invalid.

Managing Hardware Contexts

The number of reassembly contexts dedicated to messages over a given RapidIO flow priority can be limited using the Reassembly Context Threshold Registers (MURCAR0-1). Messages over flows whose corresponding threshold register is zero draw from a generic context pool determined by the generic reassembly context assignment field in MURCAR2. Contexts left over after dedication to a flow or the generic pool are in the unassigned pool. Contexts cannot be allocated by transaction type.

As shown in Figure 3, hardware maintains an open counter per flow and a generic counter to track the number of open contexts. The appropriate counter is incremented when the first segment of a multi-segment Type 9 or 11 message is classified. It is decremented when the last segment of a message is classified. If a counter reaches the configured threshold, no further new single or multi-segment messages at that flow level will be accepted and subsequent Type 10 or 11 message segments are logically retried and Type 9 messages are dropped as shown in Figure 1. This implies that Type 9 packets are dropped only if the number of interleaved messages exceeds the HW contexts allocated by the system.

By default, 16 contexts are dedicated to the generic pool and 8 are unassigned. In this configuration, the hardware will accept a maximum of 16 interleaved multi-segment messages regardless of their flow priority. The 17th Type 9 or 11 multi-segment message will be dropped or retried respectively.

When the last (or only) segment of a message is processed, it causes the HW Context to transition quickly to Invalid as descriptor writes are issued. Under normal operating conditions, when the final segment of a message is followed immediately by the first segment of a new one, the HW context used by the first message is eligible for reuse by the second. As a result, one or more streams of single or multi-segment messages whose message segments are never interleaved require only one reassembly context.

Single Segment Message Streams

A single-segment message is treated just the same as a multi-segment message in many ways. Each message still constitutes a new reassembly context and hardware still checks the flow priority of the message against the associated open counters and thresholds. If the threshold was reached before it arrives, the single-segment message exceeds the limit of interleaved contexts and a Type 11 packet is retried and a Type 9 packet is dropped.

Hence, when a stream of single-segment messages is expected, a single HW context must be reserved either by flow threshold or as part of the generic pool just as any stream of multi-segment messages. This guarantees that other interleaved multi-segment message streams do not cause the open counter to reach threshold and block the single-segment message stream.

In one sense, single-segment messages do represent a special case for context management. Single-segment message segments cannot be interleaved since they have only one “segment.” This attribute is reflected in hardware by never incrementing open counters when a single-segment message is received. As a result, single-segment messages cannot push any counter to a threshold. This implies that one or more single-segment streams may be accepted indefinitely as long as at least one context is reserved for them and the previously allocated open counter does not reach the programmed threshold.

Congestion Behavior and Management

Should eMSG encounter unusual system congestion that significantly increases the latency to memory, inbound message segments that have not yet been written to memory will begin to accumulate in internal buffers. As congestion prevents issue of descriptor updates to memory, contexts can no longer be released. Eventually, link-level retry is invoked as back-pressure to avoid Type 9 packet loss.

During periods of congestion but before data buffers fill, packets are accepted normally. New multi-segment message are accepted until open counter thresholds are reached. Single segment messages are also accepted as long as the associated open counter has not reached threshold.

As noted above, each single segment Type 9-11 message allocates a context. Hence, a stream of single-segment messages represents a worst-case scenario in terms of context allocation since a context must be allocated for each packet. When congestion occurs in this situation, exhaustion of contexts will occur well before all internal buffers are freed.

A newly arriving single segment message would normally take the context freed by the previous packet. This is especially true for Type 9 transactions which do not have to wait for a logical response. During periods of congestion, contexts that would otherwise transition to the invalid state remain in Open until congestion clears. In the absence of available invalid contexts, single segment Type 9 messages would normally cause link-level retry even though internal buffers may not yet be filled. In this situation, contexts from the unassigned pool are allocated to allow the hardware to continue accepting single segment messages. In extreme situations, even this unassigned pool will be exhausted. In this case, link-level retry will begin. Contexts in the unassigned pool act as an elastic buffer during short-term congestion events by delaying the invocation of link-level retry.

To ensure maximum performance when significant numbers of back-to-back single-segment messages are expected it is recommended that the unassigned pool contain at least eight HW contexts to cover rare worst-case short-term congestion events. Failure to do so will not result in Type 9 packet loss but may result in more frequent link-level retries. Note that link retries can cause brief performance loss when higher priority streams are carried by the link.

4 Examples

The following are examples of system configurations and the resulting behavior. A stream as used below is defined as a sequence of messages sharing the same tuple as follows:

- Type 9: <SrcID, Destid, CoS, StreamID>
- Type 11: <SrcID, DestID, mbox/xmbox, letter>

Example 1

- Inbound Traffic Types
  - 8 segment interleaved multi-segment Type 11 streams over Flow A
  - 8 segment interleaved multi-segment Type 11 streams over Flow B
  - 8 segment interleaved multi-segment Type 11 streams over Flow C
  - 8 segment interleaved multi-segment Type 11 streams over Flow D
- Context allocation (default)
  - 8 unassigned for congestion management
- Thresholds (default)
  - Generic = 16
- Endpoint behavior
  - If streams for any two flow A-D are interleaved, messages are always accepted
    - Up to 16 contexts are allocated from the generic pool
  - If more than 16 streams are interleaved regardless of which Flow they are in, the 17th and subsequent interleaved Type 11 messages will be logically retried
  - Memory congestion beyond tolerance could eventually result in Type 11 logical retries, link-level retries or both depending on sequence of events and timing

Example 2

- Inbound Traffic Types
  - 8 segment interleaved multi-segment Type 9 streams over Flow A
  - 8 segment interleaved multi-segment Type 9 streams over Flow B
  - 8 segment interleaved multi-segment Type 9 streams over Flow C
  - 8 segment interleaved multi-segment Type 9 streams over Flow D
- Context allocation (default)
  - 8 unassigned for congestion management
- Thresholds (default)
  - Generic = 16
- Endpoint behavior

- If streams for any two flow A-D are interleaved, messages are always accepted
    - Up to 16 contexts are allocated from the generic pool
- If more than 16 streams are interleaved regardless of which Flow they are in, the 17th and subsequent interleaved Type 9 messages will be dropped
- Memory congestion beyond tolerance could eventually result in Type 11 logical retries, link-level retries or both depending on sequence of events and timing

Example 3

- Inbound Traffic Types
    - 23 segment interleaved multi-segment Type 11 streams over Flow A
- Context allocation
    - 23 for Flow A
    - 1 unassigned for congestion management
- Thresholds
    - Flow A = 23
    - Generic = 0
- Endpoint behavior
    - All segments of Type 11 message streams 1-23 always accepted
    - Even though only 1 context is allocated for congestion management, configuration is tolerant of memory congestion because Type 11 multi-segment streams are more tolerant of congestion
    - Memory congestion beyond tolerance could eventually result in Type 11 logical retries, link-level retries or both depending on sequence of events and timing

Example 4

- Inbound Traffic Types
    - 30 segment interleaved multi-segment Type 11 streams over Flow A
- Context allocation
    - 23 for Flow A
    - 1 unassigned for congestion management
- Thresholds
    - Flow A = 23
    - Generic = 0
- Endpoint behavior
    - Messages from streams 1-30 may be accepted depending on the order of the message and segments
    - Segments from up to 23 messages will be accepted at any given moment
    - All others will be logically retried

        - ▪ No control at the receive endpoint over which streams are accepted and which are retried is possible
    - o Even though only 1 context is allocated for congestion management, configuration is tolerant of memory congestion because Type 11 multi-segment streams are more tolerant of congestion
        - ▪ Memory congestion beyond tolerance will eventually result in Type 11 logical retries, link-level retries or both

Example 5

- Inbound Traffic Types
    - o 23 segment interleaved multi-segment Type 9 streams over Flow A
- Context allocation
    - o 23 for Flow A
    - o 1 unassigned for congestion management
- Thresholds
    - o Flow A = 23
    - o Generic = 0
- Endpoint behavior
    - o All segments of Type 9 message streams 1-23 always accepted
    - o Even though only 1 context is allocated for congestion management, configuration is tolerant of memory congestion because multi-segment streams are more tolerant of congestion
    - o Memory congestion beyond tolerance will eventually result in link-level retries

Example 6

- Inbound Traffic Types
    - o 30 segment interleaved multi-segment Type 9 streams over Flow A
- Context allocation
    - o 23 for Flow A
    - o 1 for congestion management
- Thresholds
    - o Flow A = 23
    - o Generic = 0
- Endpoint behavior
    - o Various messages from Type 9 streams 1-30 may be accepted depending on the order of the message and segments
    - o No more than 23 messages will be reassembled at one time
    - o All other messages will be dropped

- No control at the receive endpoint over which streams are accepted and which are dropped is possible

Example 7

- Inbound Traffic Types
  - 16 segment interleaved multi-segment Type 11 streams over Flow A
  - 1 single-segment Type 9 stream over Flow A
- Context allocation
  - 16 for Flow A
  - 8 unassigned for congestion management
- Thresholds
  - Flow A = 16
  - Generic = 0
- Endpoint behavior
  - All segments of Type 11 message streams 1-16 always accepted
  - Type 9 segments are dropped if they arrive when all 16 Type 11 streams have outstanding segments (i.e. all 16 contexts are in Open state thus causing the Threshold for Flow A to be reached)

Example 8

- Inbound Traffic Types
  - 8 interleaved single-segment Type 9 streams over Flow A
  - 8 interleaved single-segment Type 9 streams over Flow B
- Context allocation
  - 8 for Flow A
  - 8 for Flow B
  - 8 unassigned for congestion management
- Thresholds
  - Flow A = 8
  - Flow B = 8
  - Generic = 0
- Endpoint behavior
  - All segments of Type 9 message streams 1-8 always accepted over Flow A
  - All segments of Type 9 message streams 1-8 always accepted over Flow B
  - Tolerant of memory congestion because recommended 8 entries are allocated for congestion management
  - Memory congestion beyond tolerance will eventually result in link-level retries

Example 9

- Inbound Traffic Types
  - 15 segment interleaved multi-segment Type 11 streams over Flow A
  - 16 single segment Type 9 streams over Flow B
- Context allocation
  - 15 for Flow A
  - 1 for Flow B
  - 8 unassigned for congestion management
- Thresholds
  - Flow A = 15
  - Flow B = 1
  - Generic = 0
- Endpoint behavior
  - All segments of Type 11 message streams 1-15 always accepted
  - All segments of Type 9 message streams 1-16 always accepted
  - Only a single context is required for one or more streams of single segment messages
  - Tolerant of memory congestion because recommended 8 entries are allocated for congestion management
  - Memory congestion beyond tolerance will eventually result in Type 11 logical retries, link-level retries or both

Example 10

- Inbound Traffic Types
  - 11 segment interleaved Type 11 streams over Flow A where each stream is a mix of single and multi-segment messages
  - 5 segment interleaved Type 9 streams over Flow B where each stream is a mix of single and multi-segment messages
- Context allocation
  - 11 for Flow A
  - 5 for Flow B
  - 8 unassigned for congestion management
- Thresholds
  - Flow A = 11
  - Flow B = 5
  - Generic = 0
- Endpoint behavior
  - All segments of Type 11 message streams 1-11 always accepted

- All segments of Type 9 message streams 1-5 always accepted
- Tolerant of memory congestion because recommended 8 entries are allocated for congestion management
- Memory congestion beyond tolerance will eventually result in Type 11 logical retries, link-level retries or both

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +303-675-2140
Fax: +303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: XX0000
Rev. X
09/2005