1 Introduction

The MPC5510 family of 32-bit microcontrollers is Freescale Semiconductor's latest achievement in integrated automotive application controllers.

The on-chip FlexCAN modules have been taken from the MPC5500 family and modified to provide some powerful new features while obtaining backwards compatibility with older CAN modules.

There are up to six enhanced full CAN (FlexCAN) modules with configurable buffers on MPC5510 microcontrollers. Each FlexCAN module is a communication controller implementing the CAN protocol according to Bosch Specification version 2.0B and ISO Standard 11898.
New Module Features

The FlexCAN module has been adapted to include some powerful new features and allow backwards compatibility with older CAN modules. These new features are:

- Full featured Rx FIFO
  - Storage capacity for six frames and internal pointer managing
  - Powerful Rx FIFO ID filtering, capable of matching incoming IDs against eight extended, 16 standard, or 32 partial (8 bits) IDs, with individual masking capability
- Additional local priority programmable transmission on individual Tx message buffers.
- Hardware cancellation on Tx message buffers.

These new features have selectable backwards compatibility with previous FlexCAN versions.

2 New Module Features

This section describes and discusses each of the new module features and examples of how to use these features.

2.1 Rx FIFO

When the FIFO Enable (FEN) bit is set in the CANx_MCR (see Appendix A), the memory area from 0x80 to 0xFF (which is normally occupied by MBs 0 to 7) is used by the reception FIFO engine.

Figure 1 shows the start of the memory structure for the message buffers when FEN equals 0.

Figure 2 shows the Rx FIFO data structure when FEN equals 1.

- Region 0x0 – 0xC contains an MB structure that is the port which the CPU reads data through from the FIFO (the oldest frame received and not read yet).
- The region 0x10 – 0xDF is reserved for internal use of the FIFO engine.
- The region 0xE0 – 0xFF contains an eight-entry ID table that specifies filtering criteria for accepting frames into the FIFO. Figure 3 shows the three different formats that the elements of the ID table can assume, depending on the IDAM field of the CANx_MCR. All elements of the table must have the same format.

This allows the CPU to read the received frames sequentially, in the order they were received, by repeatedly accessing a message buffer structure at the beginning of the memory.

NOTE

This means that message buffers 0-7 can not be used as standard Tx or Rx buffers.
New Module Features

Figure 1. Default Memory Buffer Structure

Figure 2. Rx FIFO Memory Structure

MPC5510 New FlexCAN Module Features, Rev. 0

Freescale Semiconductor
Perform the following steps to use the Rx FIFO feature:

1. Set the FEN bit in the CANx_MCR.
2. Set the IDAM field in the CANx_MCR (see Figure 4).
3. Set the ID table entries with the appropriate IDs, depending on the format chosen in step 2.
4. Enable the CAN module for transmission and/or reception (after other initialization and configuration).
5. Allow polling of the FIFO interrupt flags (or set up interrupt routine if enabled) for FIFO activity:
   — BUF5I - Buffer MB5 Interrupt or Frames available in FIFO.
   — BUF6I - Buffer MB6 Interrupt or FIFO Warning. Four frames have accumulated in the FIFO.
   — BUF7I - Buffer MB7 Interrupt or FIFO Overflow. The FIFO is full and subsequent frames are not accepted until the CPU creates space in the FIFO by reading one or more frames. While the FIFO is full, the frames are only received if they are matched with another MB (MB8-MB63).
6. Read the frame (top FIFO entry).
7. Clear the FIFO interrupt flag. The act of clearing the interrupt triggers the FIFO engine to replace the MB with the next frame in the queue and then issues another interrupt to the CPU.

The FIFO can be disabled by clearing the FEN bit in the CANx_MCR when finished.

### Figure 4. IDAM Coding

<table>
<thead>
<tr>
<th>IDAM</th>
<th>Format</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>A</td>
<td>One full ID (standard or extended) per filter element.</td>
</tr>
<tr>
<td>01</td>
<td>B</td>
<td>Two full standard IDs or two partial 14-bit extended IDs per filter element.</td>
</tr>
<tr>
<td>10</td>
<td>C</td>
<td>Four partial 8-bit IDs (standard or extended) per filter element.</td>
</tr>
<tr>
<td>11</td>
<td>D</td>
<td>All frames rejected.</td>
</tr>
</tbody>
</table>
The header file available from Freescale supports this new feature. If this is not used, the ID table can be declared as follows (CAN_A example):

```c
#define ID_Table0_CANA (*((uint32_t* )(0xfffc00e0))
#define ID_Table1_CANA (*((uint32_t* )(0xfffc00e4))
#define ID_Table7_CANA (*((uint32_t* )(0xfffc00fC))
```

Without the Rx FIFO, receiving frames could involve message buffers having a high priority to read the data and then re-open the message buffer for the next frame. If the priority was not high enough or the CPU could not service the interrupt fast enough, it could result in Rx frames being overwritten. The Rx FIFO overcomes this problem by giving the CPU more time to service the message buffers, allowing up to six messages to be stored, giving more flexibility.

### 2.2 Local Priority Transmission

The term local priority refers to the priority of transmit messages of the host node. This allows increased control over the priority mechanism for transmitting messages. Figure 5 shows the placement of PRIO in the ID part of the message buffer.

An additional 3-bit field (PRIO) in the long-word ID part of the message buffer structure has been added for local priority determination. They are prefixed to the regular ID to define the transmission priority.

These bits are not transmitted and are intended only for Tx buffers.

![Figure 5. Message Buffer Structure](image)

Perform the following to use the local priority feature:

1. Set the LPRIO_EN bit in the CANx_MCR.
2. Write the additional PRIO bits in the ID long-word of Tx message buffers when configuring the Tx buffers.

See Appendix B for a code example on configuring the Tx buffers and setting the local priority bits.

With this extended ID concept, the arbitration process is based on the full 32-bit word. However, the actual transmitted ID continues to have 11 bits for standard frames and 29 bits for extended frames.

### 2.3 Hardware Cancellation (Transmission Abort Mechanism)

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be
aborted and was transmitted instead. This feature allows transmission to be aborted for the message buffer to be updated.

Set the AEN bit in the CANx_MCR to enable the transmission abort mechanism.

The CPU must then write a specific abort code (1001) to the code field of the control and status word of the Tx message that needs to be aborted.

If the abort code is written to an MB currently being transmitted or to an MB already loaded into the SMB (serial message buffer) for transmission, the write operation is blocked and the MB is not deactivated. However, the abort request is captured and kept pending until one of the following conditions are satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into freeze mode

The abort procedure can be summarized as follows:

- CPU writes 1001 (binary) into the code field of the C/S word
- CPU reads the CODE field and compares it to the value that was written
- If the CODE field that was read is different from the value that was written, the CPU must read the corresponding IFLAG to check if the frame was transmitted or it is being currently transmitted. If the corresponding IFLAG is set, the frame was transmitted. If the corresponding IFLAG is reset, the CPU must wait for it to be set, and then the CPU must read the CODE field to check if the MB was aborted (CODE=1001) or transmitted (CODE=1000).

As an application example, this could avoid sending out-of-date data.

For example, if the data to be transmitted has been updated before the corresponding message is pending transmission, it can be aborted. Then, the data can be updated and the transmission sequence can restart. This is controlled by the application software.

3 Conclusion

These three current features added to the FlexCAN modules allow increased flexibility and backwards compatibility with all CAN modules.

- The Rx FIFO allows a powerful filtering mechanism for Rx messages, with CPU interrupting.
- The local priority mechanism allows extended IDs for Tx messages.
- The hardware cancellation allows transmission to be aborted for the message buffer to be updated.

These additional features are exclusive to Freescale Semiconductor's FlexCAN module on the MPC5510 family of microcontrollers.
Appendix A
CANx Module Configuration Register (CANx_MCR)

The CANx Module Configuration Register (CANx_MCR) is mentioned numerous times throughout this application note. This is where the main controls are for enabling these new features, which allows backward compatibility.

- FEN - FIFO Enable
  - IDAM - ID Acceptance Mode.
- AEN - Abort Enable
- LPRIO_EN - Local Priority Enable

Appendix B
Configuring Tx Buffers Code Example

This code example shows a basic function to configure the message buffers and includes local priority setting.

```c
void CAN_Tx_start (UINT8 data, UINT8 data_inc, UINT8 length, UINT8 buf) {
    UINT16 i;
    UINT16 timer;

    CAN_A.BUF[buf].DATA.W[0] = 0x00000000;
    CAN_A.BUF[buf].DATA.W[1] = 0x00000000; // Clear Data fields
    CAN_A.BUF[buf].CS.B.CODE = 0x8; // Hold the transmit buffer inactive
    CAN_A.BUF[buf].ID.B.PRI0 = can.ARX.NumMsgs>>1; // Set Local Priority
    CAN_A.BUF[buf].ID.B.STD_ID = can.ATX.ID; // Write standard MB IDs
    for (i=0; i < length; i++){
        CAN_A.BUF[buf].DATA.B[i] = data; // Write data
        data += data_inc;
    }
    CAN_A.BUF[buf].CS.B.LENGTH = length; // Write length
    CAN_A.BUF[buf].CS.B.CODE = 0xC; // Write Code (C = Transmit)
    timer = CAN_A.TIMER.R; // Unlock Message buffers
}
```
Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals", must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized use, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

RoHS-compliant and/or Pb-free versions of Freescale products have the functionality and electrical characteristics as their non-RoHS-compliant and/or non-Pb-free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale’s Environmental Products program, go to http://www.freescale.com/epp.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. © Freescale Semiconductor, Inc. 2007. All rights reserved.