

PREFACE

The *BestComm API User's Guide* describes the use of the BestComm AP .

This manual is organized as follows:

- Section 1 gives an overview of the BestComm AP .
- Section 2 describes the BestComm API functions.
- Section 3 describes the tasks included with the BestComm API release.
- Section 4 describes how to use the BestComm API and gives some sample code examples.
- Appendix A describes the tasks that are included with the BestComm AP release in more detail.
- Appendix B contains the generated documentation details for the BestComm API.

TABLE OF CONTENTS

SECTION	PAGE NUMBER
Section 1	
Overview	
1.1	Introduction 1-1
1.1.1	MPC5200 BestComm DMA Engine 1-1

1.2 BESTCOMM API

SECTION 2

USING THE BESTCOMM API


```
* with the sdma register location after the taskba_ has been
* loaded with the destination address of the task image. In
* this instance, the address is the beginning of SRAM.
*/
sdma = (sdma_regs *) (MBAR + SDMA_REG_OFFSET);
sdma->taskBa_ = MBAR + SRAM_OFFSET;
```

```

}

/*
 * The interrupt routine clears the interrupt.
 */
int main_interrupt_routine(void *arg0, void *arg1)
{
#pragma unused (arg0, arg1)
    /*
     * Check to see if task called the interrupt. The return for
     * TaskIntStatus is the task number if an interrupt is
     * pending.
     */
    if(TaskIntStatus(PSC1TXTaskId) == PSC1TXTaskId)
    {
        /*
         * Clear the task interrupt
         */
        TaskIntClear(PSC1TXTaskId);

        /*
         * Transfer is finished
         */
        interrupted = 1;
    }
    return 1;
}

```

2.3 BUFFER DESCRIPTOR TASKS

Figure 2-3. Buffer descriptor task program flow

2.3.1 Buffer Descriptor Sample Code

The following sample code shows how to use the API with a buffer descriptor task. This

* /

Figure 2-5. Virtual memory and multi-process program flow

2.4.1 Virtual Memory and Multi-Process Sample Code

Using the BestComm API

```
/* Call TasksGetSramOffset() to get the free SRAM address after the tasks. */
```



```
}  
  
}
```


TasksGetSramOffset(uint32 sram_offset)

This function returns the offset set aside in the sram by the API. The sample code illustrates how to use this function.

4.1.4 Task Related Functions

*TaskSetup(TaskName_t TaskName, TaskSetupParamSet_t *TaskParams)*

The *TaskSetup()* function prepares a task for use. The TaskName is the name of the task to use. The TaskParams structure should be filled in before calling this function. This function returns a handle to the task to be used by other API functions.

TaskStart(TaskId taskId, uint32 autoStartEnable, TaskId autoStartTask, uint32 intrEnable)

This function starts the task represented by taskId with the option to auto start another task or the same task when done. The task ca8funf(0mme)8.6(sk)10t10.6(d)-1.4(si1.4(st o)8.k w)4

4.1.6 Buffer Descriptor Related Functions

*TaskBDAssign(TaskId taskId, void *buffer0, void *buffer1, int size, uint32 bdFlags)*

This function assigns a buffer descriptor to the buffer ring for the buffer descriptor task represented by taskId.

TaskBDRelease(TaskId taskId)

This function removes the last buffer descriptor that was used from the buffer ring for the task represented by taskId.

TaskGetBD(TaskId taskId, BDIdE bd)

Table 4-1. Initiator Definitions

Enumeration values

INITIATOR_ALWAYS
INITIATOR_SCTMR_0
INITIATOR_SCTMR_1
INITIATOR_FEC_RX
INITIATOR_FEC_TX
INITIATOR_ATA_RX
INITIATOR_ATA_TX
INITIATOR_SCPCI_RX
INITIATOR_SCPCI_TX

SzDst - Transfer size in bytes (1, 2 or 4)

Buffers are added to the buffer descriptor ring by calling TaskBDAssign(). A call to TaskBDRelease() removes a buffer from the ring. The first buffer pointer passed to TaskBDAssign() is used as the destination address base pointer. The second buffer pointer

