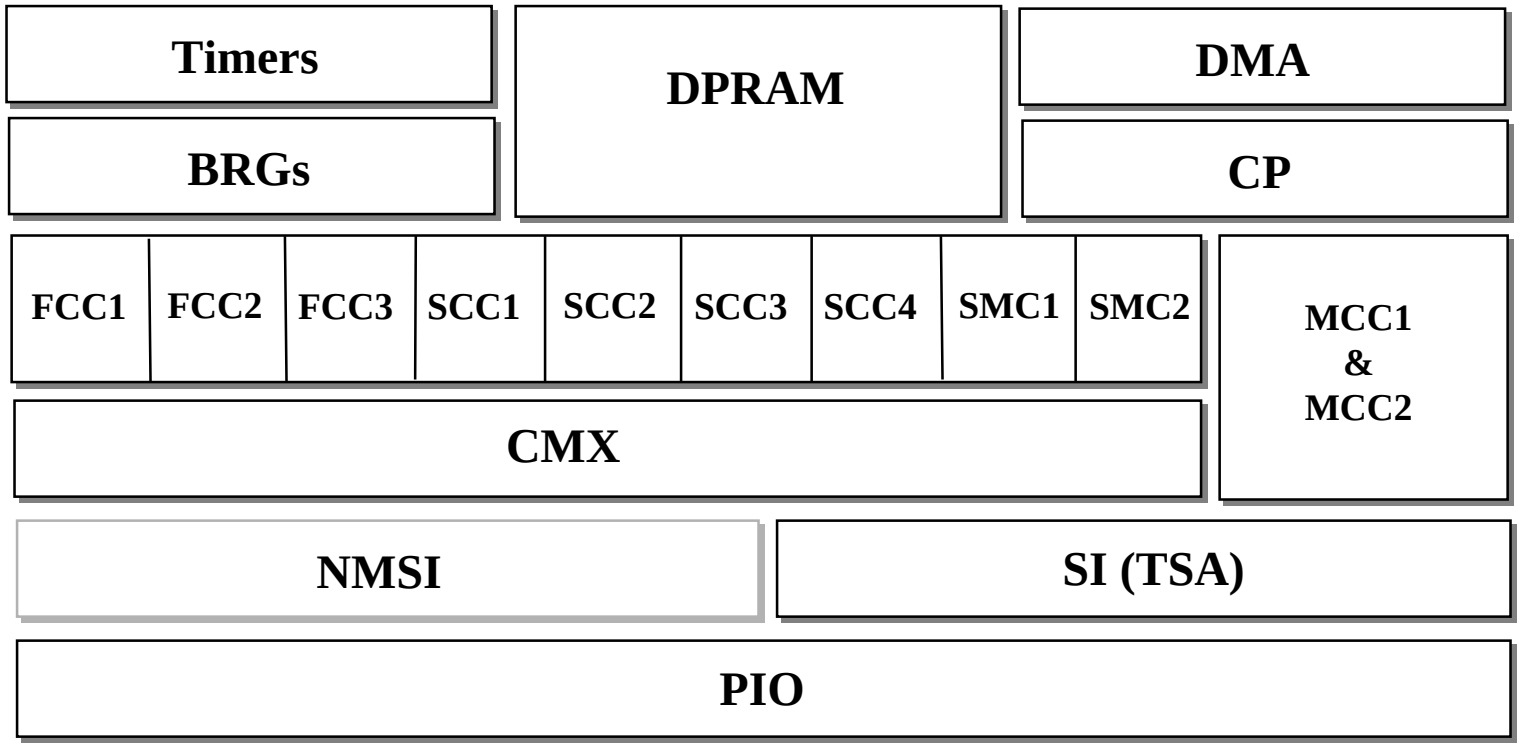
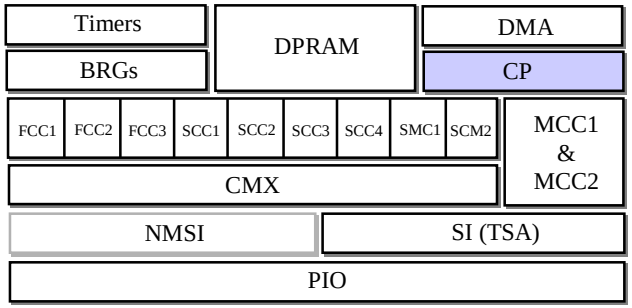
# MSC8101 Communications Processor Module: An Introduction

Version 1 : Date: 3/7/2002

# CPM Programming Model

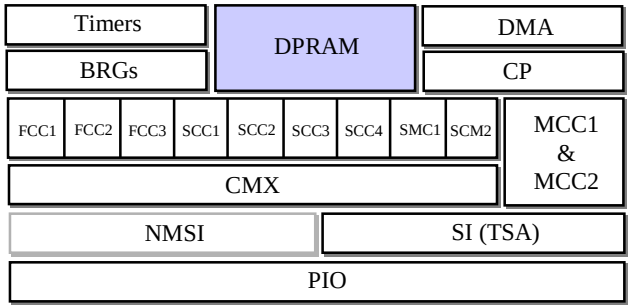


# CP - Communications Processor



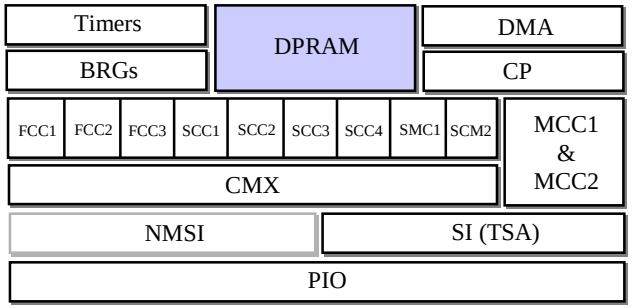

- **32-bit RISC processor**
- **Controls lower layer protocols independent of the core**
- **Manages data transfer between I/O and memory**
- **Manages up to 16 software timers**
- **Executes code from internal ROM or RAM**
- **The core issues CP commands by writing CPCR**

# DPRAM - Dual-port RAM



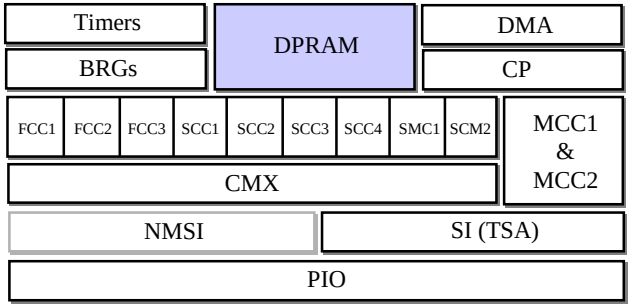

- **DPRAM is used to store:**
  - **Parameters for communication controllers and DMA**
  - **Buffer descriptors (BDs)**
  - **Data buffers**
  - **CP microcode**
  - **Temporary data for FCC FIFO**
  - **Scratch memory**
- **DPRAM is accessed by:**
  - **CP**
  - **Core**
  - **SDMA over 60x bus**
  - **SDMA over local bus**

# Parameter RAM

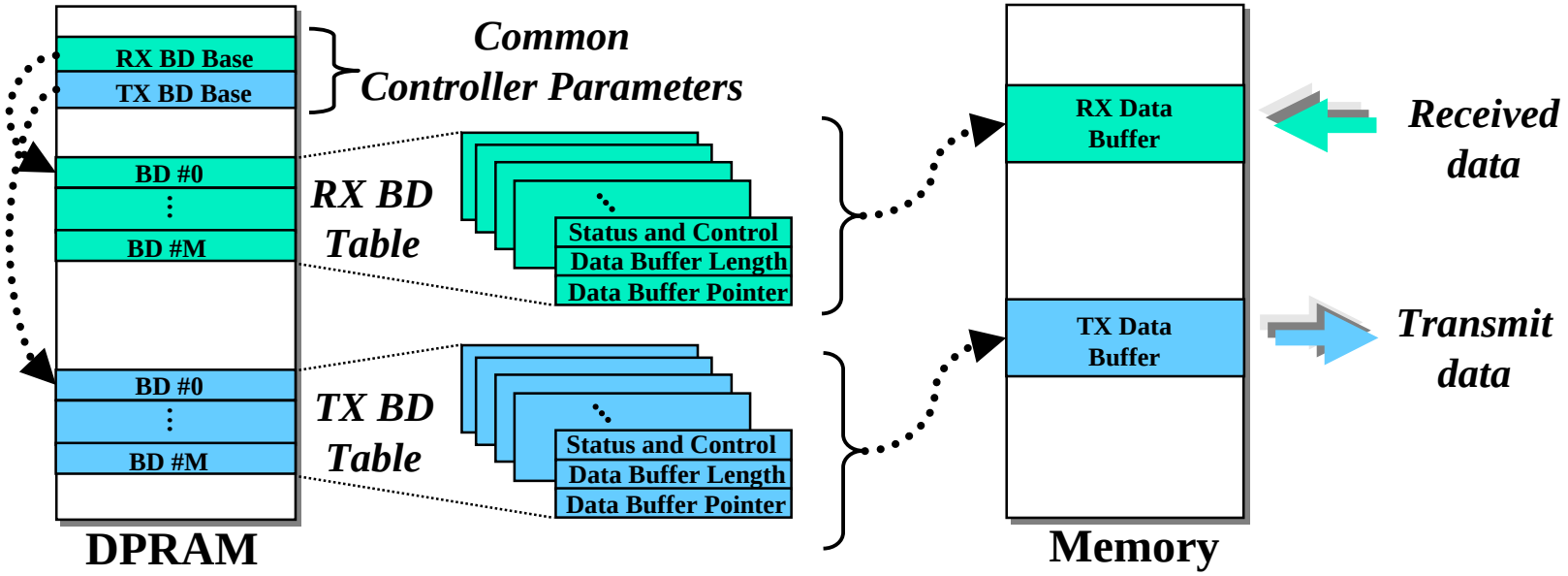



- **Used to store parameters for the operation of**
  - **FCCs SCCs SMCs SPI $I^2C$ IDMA**
- **Types of parameters**
  - **Common to a controller**
  - **Specific to a protocol**
- **Parameter RAM access by the core**
  - **RX and TX parameters can be read at any time**
  - **RX parameters written only when receiver is disabled**
  - **TX parameters written only when transmitter is disabled**

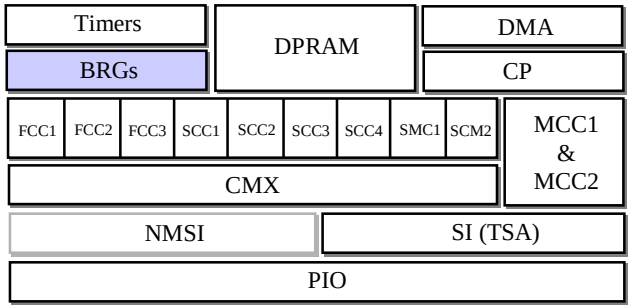
# Buffer Descriptors (BDs) and Data Buffers



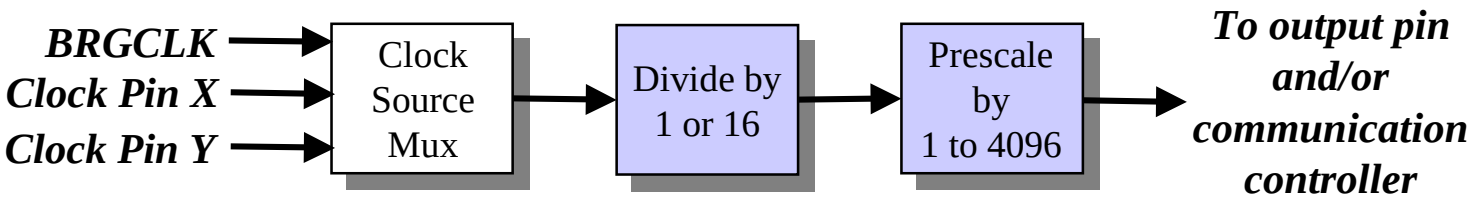

- **Buffer descriptors are always used by communication controllers to control access in/out of data buffers**
- **All RX and TX BDs have the same 8-byte format**



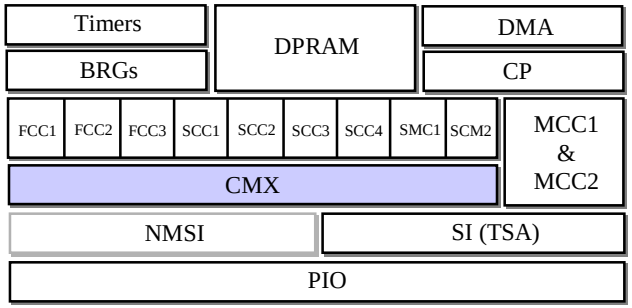
# BRGs - Baud Rate Generators




- **Eight independent and BRG's can be routed to one or more**
  - **FCCs SCCs SMCs Output pins**
- **The BRG's clock can come from**
  - **BRGCLK which is generated internally by the PLL**
  - **One of two external clock pins**



# CMX - CPM Multiplexing



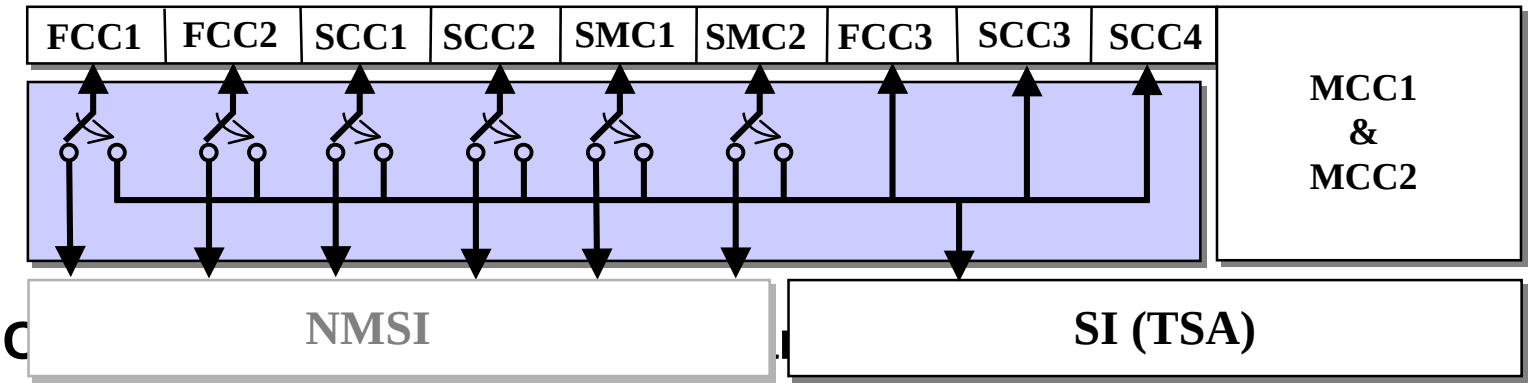

- **Connects the communication controllers**

  **FCCs SCCs SMCs**

  **to the physical layer**

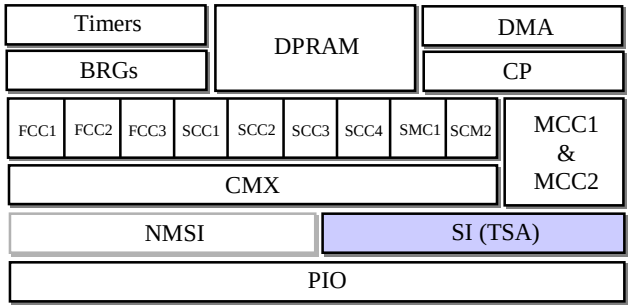
  **NMSI: UTOPIA, MII, modem lines, proprietary serial lines**

  **and SI: TDM channels**



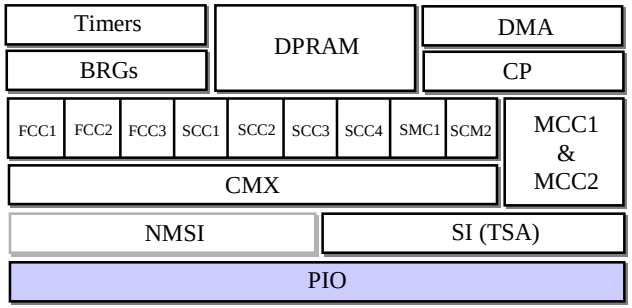

- C

# SI - Serial Interface and Time-slot Assigner




- **Two SIs implement routing and time-division multiplexing**
  - **SI1: TDMA1**
  - **SI2: TDMA2, TDMB2 and TDMC2**
- **Each SI has a receive and a transmit RAM**
  - **Independent of DPRAM and accessible by the core**
  - **Each TDM is assigned a portion of the SI RAM**
  - **Defines: time-slot size and location within a frame**
    - **routing with communication controllers**
    - **behavior of the strobe outputs**
- **Static and Dynamic routing supported**
- **Loop-back and echo modes available for testing**

# PIO - Parallel I/O




- **MSC8101 CPM supports four ports**

  **Port A: 26 pins  Port B: 14 pins  Port C: 18 pins  Port D: 8 pins**

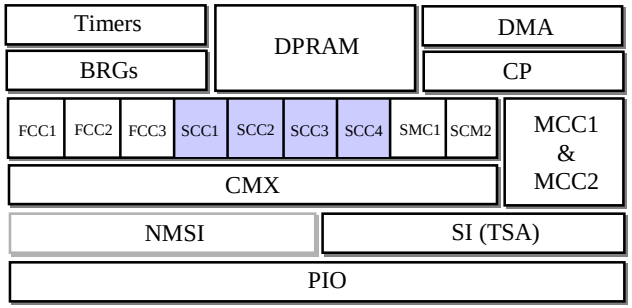
- **Port pins can be configured as:**

  **Bi-directional general-purpose I/O**
  **Dedicated peripheral interface signal**

- **Functional operation**

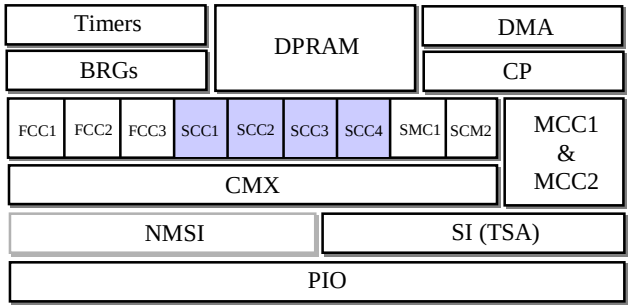
| Register | Description | 0 | 1 |
|---|---|---|---|
| PPAR[A-C] | Port pin assignment | General-purpose | Dedicated |
| PSOR[A-C] | Special operation | Option 1 | Option 2 |
| PDIR[A-C] | Pin direction | Input | Output |
| PODR[A-C] | Ouput drive | Regular | Open drain |
| PDAT[A-C] | Pin data | 0 | 1 |

# SCC - Serial Communication Controller




- **An SCC is described in terms of the protocol it runs**
  - **Implements parts of the link layer (OSI Layer 2) protocols**
  - **The SCC and the protocol are independent of the physical interface**
- **Protocols**
  - **UART, HDLC, AppleTalk/LocalTalk, BISYNC, Ethernet, and transparent**
- **Synchronous and asynchronous operation**
- **Specific steps required to:**
  - **Reconfiguring receiver or transmitter**
  - **Changing protocols for receiver or transmitter**
  - **Servicing interrupts**

# SCC - Serial Communication Controller (cont.)




- **Setting up an SCC:**
  1. **Initialize PIO**
  2. **Initialize SI[1-2] (if used)**
  3. **Initialize BRG (if used)**
  4. **Initialize CPM mux**
  5. **Initialize general SCC[1-4] registers**
  6. **Initialize protocol specific SCC[1-4] registers**
  7. **Initialize SCC[1-4] parameter RAM**
  8. **Issue command to CP**
  9. **Clear prior events**
  10. **Enable interrupts (if used)**
  11. **Enable receiver and/or transmitter**

# SCC UART

- **Used for low speed data between terminal devices**
- **Protocol-specific status and control information**
  - **BD control and status field**
  - **Error counters in UART parameter RAM**
  - **Event register (SCCE)**
- **Asynchronous mode:**
  - **1 start bit**
  - **5-8 data bits (lsb first)**
  - **address or data bit (optional)**
  - **parity bit (optional)**
  - **stop bit (which can be fractional)**

# SCC UART Character Echo Example

- **SCC1 UART Initialization**
  1. **Initialize PIO**
     **PSORC, PSORD, PPARC, PPARD,**
     **PDIRC, PDIRD, PODRC, PODRD**
  2. **Initialize SI[1-2] (if used)**
  3. **Initialize BRG (if used)**
     **BRGC1**
  4. **Initialize CPM mux**
     **CMXSCR**
  7. **Initialize SCC1 parameter RAM**
     **RxBD and TxBD,**
     **Parameter RAM: RFCR, TFCR, RBASE, TBASE, MRBLR**
  5. **Initialize general SCC registers**
     **GSMR_H, GSMR_L**

# SCC UART Character Echo Example (cont.)

- **SCC1 UART Initialization (cont.)**
  6. **Initialize protocol specific SCC registers**
     - **PSMR**
  8. **Issue command to CP**
     - **CPCR**
  9. **Clear prior events**
     - **SCCE**
  10. **Enable interrupts (if used)**
  11. **Enable receiver and/or transmitter**
     - **GSMR_L**

# SCC UART Character Echo Example

- **SCC1 UART Buffer Descriptors**

```
BDRINGS *RxTxBD;     /* BD base pointer */
```

```
typedef struct BufferDescRings
{
    BD RxBD;     /* Rx BD ring */
    BD TxBD;     /* Tx BD ring */
} BDRINGS;
```

```
typedef struct BufferDescriptor
{
   UHWORD bd_cstatus;      /* control and status */
   UHWORD bd_length;       /* transfer length */
   UBYTE  *bd_addr;        /* buffer address */
} BD;
```

- **SCC1 UART Data Buffers**

```
LB *SCC1Buffers;     /* SCC1 base pointers */
```

```
typedef struct BufferPool
{
   UBYTE  RxBuffer;
   UBYTE  TxBuffer;
} LB;
```

# SCC UART Character Echo Example

- **SCC1 UART main program**

```
void main(void)
{

IMM =  (t_PQ2IMM *)0x04700000;  /* MPC8260 internal register map  */
RxTxBD = (BDRINGS *)(((UWORD)&(IMM->pram.serials.scc_pram[SCC1])) + 72);
SCC1Buffers = (LB *)(((UWORD)&(IMM->pram.serials.scc_pram[SCC1])) + 96);

InitSCC1Uart();

   while (1)
   {
      if (SCC1Poll())   /* Check BD status for Rx characters */
      {
         EchoChar();
      }

      BlinkSignalingLED();   /* Blink the Signaling LED */
   }

} /* END main */
```