Evaluating the MC68360 Serial Performance

By Kevin P. Godfrey,

April 13, 1994

The MC68360 Quad Integrated Communications Controller (QUICC) has many seven controllers and it is often important to evaluate the maximum serial performance of these controllers in various configurations. This note explains how to test the QUICC's serial performance in various configurations and gives the software required for the testing.

The QUICC includes four general purpose serial communication controllers (SCCs), two serial management controllers (SMCs) and a serial peripheral interface (SPI). These are all under the control of a RISC Communication Processor (CP). The CP also controls sixteen RISC timers and the Parallel Interface Port (PIP).

When more than one of these peripherals are operating, events requiring CP activity occur asynchronously and irregularly. Therefore, when systems are designed that are likely to heavily load the CP, it is best to test the CP performance with the final applications. This is not usually possible since the application software and hardware are still under design but is it possible to simulate the application on a QUICC development platform. This note gives a simple simulation of an application using the QUADS development board where four SCCs are running in HDLC mode.

For more information on the QUICC and QUADS, please refer to the MC68360 User's Manual and QUADS manuals.

Evaluating the CP Load

When multiple requests arrive at the CP to be serviced, they are handled in the following priority order:

Highest	1.	Reset in CP command register or System Reset
	2.	SDMA bus error
	3.	Commands issued in the CP command register
	4.	SCC1 reception
	5.	SCC1 transmission
	6.	SCC2 reception
	7.	SCC2 transmission
	8.	SCC3 reception
	9.	SCC3 transmission
	10.	SCC4 reception
	11.	SCC4 transmission
	12.	SMC1 reception
	13.	SMC1 transmission
	14.	SMC2 reception
	15.	SMC2 transmission
	16.	SPI reception
	17.	SPI transmission
	18.	PIP
Lowest	19.	RISC Timers

Under extreme conditions, too many requests can arrive at the CP in a given period for it to

handle. In this case, the lower priority requests are ignored. When this is a serial channel, its FIFOs will underrun or overrun and cause problems elsewhere in the system.

The RISC timers are the first source of requests to be ignored. If the CP is too busy to service the RISC timer requests regularly, they will count slower than a hardware timer or, in extreme cases, stop altogether. This phenomena can be used to test the loading on the CP. By using suitable software, the RISC timers can be compared to a hardware timer running at the same speed. If the RISC and hardware timers are started at the same time, their counts should always be within one counter tick of each other whenever they are sampled.

When all sixteen RISC timers are operating, they use approximately 4% of the CP performance. If other CP activity exceeds a 96% utilisation of the CP, the RISC counters will not increment regularly. When performance testing is complete and the RISC timers are not running, there is approximately 4% spare capacity in the CP to handle other requests.

To simulate an application, the serial channels must be running at their worst case traffic loading. The CP loading can be increased by transmitting and receiving short messages while using data buffers shorter than the messages forcing the CP into constantly switch buffers.

Further details of the RISC timers and how they can be used to test CP performance are given in the QUICC User's manual, section 7.4.

Software to Test the CP Load

Three software routines are given at the end of this note to test the CP loading. The routines do the following:

PERFINIT	Initialises the RISC timers.
CHKPERF	Read and print the RISC timer counter and hardware timer counter values.
SCCINIT	Initialise the four SCCs to simulate an application.

To test serial performance, run PERFINIT to initialise the RISC and hardware timers. Then start an application using the serial channels, in this example SCCINIT simulates a basic application. Once the application has completed, run CHKPERF to read and print the timer counter values. If the counters differ by more that one, then the CP is overloaded and in danger of losing data.

The software is written for use on the QUADS board but can easily be ported to any application board. The CHKPERF routine uses a QUADS system call to print the register values to the QUADS terminal.

Initialising the RISC and Hardware Timers

The PERFINIT routine initialises and enables all sixteen RISC timers to count 16K system clock ticks. The CP maintains a count of the 16K system clock ticks in the TM_cnt parameter. The PERFINIT routine also configures hardware Timer 1 to count 16K system clock ticks by cascading Timers 1 and 2.

The PERFINIT routine should only be run once to configure and enable the timers.

Reading the Timers

The CHKPERF routine should be run once the application is running or on its completion. CHKPERF reads and prints the values of the RISC timer counter and Timer 1. The user can then

verify that the RISC timer has not missed counter ticks due to CP overloading.

A Sample Application

The SCCINIT routine initialises and enables the QUICC's four SCCs in HDLC mode at a low data rate (64Kbps on a QUICC running at 25Mhz). SCCINIT provides the minimum initialisation required to operate the SCCs in loopback mode. SCCINIT initialises each SCC to have a separate clock source and uses single data buffers in continuous mode. Once SCCINIT has been run, all four SCCs will be transmitting and receiving data in loopback mode continuously. The user can now modify the baud rate generators to vary the SCC data rates.

To obtain an initial estimate of the CP loading, it is only necessary to run the sample application for a few seconds and then run CHKPERF. However, when the CP is nearing saturation level, longer periods should be allowed. It is also best to simulate the real application under worst case traffic patterns over a period of several hours.

SCCINIT can be modified to run the SCCs, SMCs, etc. at different data rates, with different protocols, with different data traffic and different data buffer sizes to simulate other applications.

A Typical Example

Table 1 shows the results of running the performance test software given in this note on a 25MHz QUICC and analysing the serial performance of four SCCs each running in HDLC mode.

Serial Clock Frequency	Value Programmed into all BRG Configuration Registers	Pass / Fail	Comment
64.1 KHz	$ 1030A	Pass	All SCCs running OK.
2.08 MHz	$ 10016	Pass	All SCCs running OK.
2.77 MHz	$ 10010	Pass	All SCCs running OK.
3.13 MHz	$ 1000E	Fail	All SCCs running but timer counters diverging slowly.
3.57 MHz	$ 1000C	Fail	All SCCs running but timer counters diverging rapidly.
5.0 MHz	$ 10008	Fail	RISC timer stopped and SCC4 indicating errors.

Table 1. Typical QUICC Serial Performance

In this example, all four SCCs are running at the same speed and the RISC timers are running. As the serial data rate is increased above 3 MHz, the timers begin to diverge. This indicates that it is unsafe to run four SCCs at greater than 3MHz. As the speed increases further, the two timer counters diverge at a greater rate and eventually the RISC timer stops completely. At this point, SCCE4 indicates transmission errors as the transmit FIFO underruns.

Performance Evaluation Software

```
*****************************************************************
*  QUICC SERIAL PERFORMANCE EVALUATION SOFTWARE                 *
*****************************************************************
*  Last Modified :  11 April 1994             Rev : 1           *
*****************************************************************
*  Filename:  perftest.src                                      *
```

```
*****************************************************************
*  <C> Copyright Motorola Inc 1994                              *
*                          *
*  Written by Kevin Godfrey, Freescale EKB.                      *
*****************************************************************
* Revision History:                                             *
*    1  Initial release.                                        *
*****************************************************************
* Associated Documentation:                                     *
*    QUICC User's Manual, 1993.                                 *
*****************************************************************
* Target System: Freescale QUADS board running QUICCbug.        *
*****************************************************************


************* EQUATES TABLE *************

* Routine Locations

PERFINIT    EQU         $440000          Performance test initialisation routine
CHKPERF     EQU         $441000          Performance check routine
SCCINIT     EQU         $442000          SCC initialisation routine


* QUADS Memory Map

DPRBASE     EQU         $20000           Start of Master QUICC internal RAM

SCC1RBUFF   EQU         $4D0000          SCC1 Rx Buffer location
SCC1TBUFF   EQU         $4D0020          SCC1 Tx Buffer location
SCC2RBUFF   EQU         $4D0040          SCC2 Rx Buffer location
SCC2TBUFF   EQU         $4D0060          SCC2 Tx Buffer location
SCC3RBUFF   EQU         $4D0080          SCC3 Rx Buffer location
SCC3TBUFF   EQU         $4D00A0          SCC3 Tx Buffer location
SCC4RBUFF   EQU         $4D00C0          SCC4 Rx Buffer location
SCC4TBUFF   EQU         $4D00E0          SCC4 Tx Buffer location


* QUICC Memory Map

REGB        EQU         DPRBASE+$1000    QUICC register base
SCC1Base    EQU         DPRBASE+$0C00    SCC1 parameter RAM page
SCC2Base    EQU         DPRBASE+$0D00    SCC2 parameter RAM page
SCC3Base    EQU         DPRBASE+$0E00    SCC3 parameter RAM page
SCC4Base    EQU         DPRBASE+$0F00    SCC4 parameter RAM page
TimerBase   EQU         DPRBASE+$0DB0    RISC Timer Parameter RAM


*   CPM & RISC Timer Registers

CR          EQU         REGB+$05C0       Command Register
RCCR        EQU         REGB+$05C4       RISC Configuration Register
RTER        EQU         REGB+$05D6       RISC Timers Event Register
RTMR        EQU         REGB+$05DA       RISC Timers Mask Register

CICR        EQU         REGB+$0540       CP Interrupt Control Register
CIPR        EQU         REGB+$0544       CP Interrupt Pending Register
CIMR        EQU         REGB+$0548       CP Interrupt Mask Register
CISR        EQU         REGB+$054C       CP Interrupt In-Service Register

*   General Purpose Timers

TGCR        EQU         REGB+$0580       Timer Global Config. Register
TMR1        EQU         REGB+$0590       Timer 1 Mode Register
TRR1        EQU         REGB+$0594       Timer 1 Reference Register
TCN1        EQU         REGB+$059C       Timer 1 Counter Register
```

```
TER1        EQU         REGB+$05B0        Timer 1 Event Register
TMR2        EQU         REGB+$0592        Timer 2 Mode Register
TRR2        EQU         REGB+$0596        Timer 2 Reference Register
TCN2        EQU         REGB+$059E        Timer 2 Counter Register
TER2        EQU         REGB+$05B2        Timer 2 Event Register

*   Port Registers

PADIR       EQU         REGB+$0550        Port A Data Direction Register
PAPAR       EQU         REGB+$0552        Port A Pin Assignment Register
PAODR       EQU         REGB+$0554        Port A Open Drain Register
PADAT       EQU         REGB+$0556        Port A Data Register
PBDIR       EQU         REGB+$06B8        Port B Data Direction Register
PBPAR       EQU         REGB+$06BC        Port B Pin Assignment Register
PBODR       EQU         REGB+$06C2        Port B Open Drain Register
PBDAT       EQU         REGB+$06C4        Port B Data Register
PCDIR       EQU         REGB+$0560        Port C Data Direction Register
PCPAR       EQU         REGB+$0562        Port C Pin Assignment Register
PCSO        EQU         REGB+$0564        Port C Special Option Register
PCDAT       EQU         REGB+$0566        Port C Data Register
PCINT       EQU         REGB+$0568        Port C Interrupt Control Register

*   Serial Interface Registers

SIMODE      EQU         REGB+$06E0        SI mode register
SIGMR       EQU         REGB+$06E4        SI global mode register
SICR        EQU         REGB+$06EC        SC clock route
SIRP        EQU         REGB+$06FE        SI RAM pointer

*   SDMA Registers

SDCR        EQU         REGB+$051E        SDMA Configuration Register
SDAR        EQU         REGB+$0520        SDMA Address Register

*   Baud Rate Generators

BRGC1       EQU         REGB+$05F0        BRG1 Configuration Register
BRGC2       EQU         REGB+$05F4        BRG2 Configuration Register
BRGC3       EQU         REGB+$05F8        BRG3 Configuration Register
BRGC4       EQU         REGB+$05FC        BRG4 Configuration Register

*   Buffer Descriptors

SCC1RBD     EQU         DPRBASE+$0470     SCC1 Rx BD location
SCC1TBD     EQU         DPRBASE+$0478     SCC1 Tx BD location
SCC2RBD     EQU         DPRBASE+$0480     SCC2 Rx BD location
SCC2TBD     EQU         DPRBASE+$0488     SCC2 Tx BD location
SCC3RBD     EQU         DPRBASE+$0490     SCC3 Rx BD location
SCC3TBD     EQU         DPRBASE+$0498     SCC3 Tx BD location
SCC4RBD     EQU         DPRBASE+$04A0     SCC4 Rx BD location
SCC4TBD     EQU         DPRBASE+$04A8     SCC4 Tx BD location

*   SCC1 Registers

GSMRL1      EQU         REGB+$0600        SCC1 General Mode Register - lower long word
GSMRH1      EQU         REGB+$0604        SCC1 General Mode Register - upper long word
PSMR1       EQU         REGB+$0608        SCC1 protocol specific mode register
DSR1        EQU         REGB+$060E        SCC1 Data Synchronisation Register
SCCE1       EQU         REGB+$0610        SCC1 event register
SCCM1       EQU         REGB+$0614        SCC1 mask register

*   SCC1 Parameter RAM

RBASE1      EQU         SCC1Base+$00      RX BD Base Address
TBASE1      EQU         SCC1Base+$02      TX BD Base Address
RFCR1       EQU         SCC1Base+$04      SDMA RX Function Code
```

```
TFCR1       EQU         SCC1Base+$05      SDMA TX Function Code
MRBLR1      EQU         SCC1Base+$06      Maximum Receive Buffer Length
CMASK1      EQU         SCC1Base+$34      CRC Constant
CPRES1      EQU         SCC1Base+$38      CRC Preset
MFLR1       EQU         SCC1Base+$46      Maximum Receive Frame Length
RFTHR1      EQU         SCC1Base+$4A      Received Frames Threshold
HMASK1      EQU         SCC1Base+$4E      HDLC Address Mask

*   SCC2 Registers

GSMRL2      EQU         REGB+$0620        SCC2 General Mode Register - lower long word
GSMRH2      EQU         REGB+$0624        SCC2 General Mode Register - upper long word
PSMR2       EQU         REGB+$0628        SCC2 protocol specific mode register
DSR2        EQU         REGB+$062E        SCC2 Data Synchronisation Register
SCCE2       EQU         REGB+$0630        SCC2 event register
SCCM2       EQU         REGB+$0634        SCC2 mask register

*   SCC2 Parameter RAM

RBASE2      EQU         SCC2Base+$00      RX BD Base Address
TBASE2      EQU         SCC2Base+$02      TX BD Base Address
RFCR2       EQU         SCC2Base+$04      SDMA RX Function Code
TFCR2       EQU         SCC2Base+$05      SDMA TX Function Code
MRBLR2      EQU         SCC2Base+$06      Maximum Receive Buffer Length
CMASK2      EQU         SCC2Base+$34      CRC Constant
CPRES2      EQU         SCC2Base+$38      CRC Preset
MFLR2       EQU         SCC2Base+$46      Maximum Receive Frame Length
RFTHR2      EQU         SCC2Base+$4A      Received Frames Threshold
HMASK2      EQU         SCC2Base+$4E      HDLC Address Mask

*   SCC3 Registers

GSMRL3      EQU         REGB+$0640        SCC3 General Mode Register - lower long word
GSMRH3      EQU         REGB+$0644        SCC3 General Mode Register - upper long word
PSMR3       EQU         REGB+$0648        SCC3 protocol specific mode register
DSR3        EQU         REGB+$064E        SCC3 Data Synchronisation Register
SCCE3       EQU         REGB+$0650        SCC3 event register
SCCM3       EQU         REGB+$0654        SCC3 mask register

*   SCC3 Parameter RAM

RBASE3      EQU         SCC3Base+$00      RX BD Base Address
TBASE3      EQU         SCC3Base+$02      TX BD Base Address
RFCR3       EQU         SCC3Base+$04      SDMA RX Function Code
TFCR3       EQU         SCC3Base+$05      SDMA TX Function Code
MRBLR3      EQU         SCC3Base+$06      Maximum Receive Buffer Length
CMASK3      EQU         SCC3Base+$34      CRC Constant
CPRES3      EQU         SCC3Base+$38      CRC Preset
MFLR3       EQU         SCC3Base+$46      Maximum Receive Frame Length
RFTHR3      EQU         SCC3Base+$4A      Received Frames Threshold
HMASK3      EQU         SCC3Base+$4E      HDLC Address Mask

*   SCC4 Registers

GSMRL4      EQU         REGB+$0660        SCC4 General Mode Register - lower long word
GSMRH4      EQU         REGB+$0664        SCC4 General Mode Register - upper long word
PSMR4       EQU         REGB+$0668        SCC4 protocol specific mode register
DSR4        EQU         REGB+$066E        SCC4 Data Synchronisation Register
SCCE4       EQU         REGB+$0670        SCC4 event register
SCCM4       EQU         REGB+$0674        SCC4 mask register

*   SCC4 Parameter RAM

RBASE4      EQU         SCC4Base+$00      RX BD Base Address
TBASE4      EQU         SCC4Base+$02      TX BD Base Address
RFCR4       EQU         SCC4Base+$04      SDMA RX Function Code
```

```
TFCR4       EQU         SCC4Base+$05      SDMA TX Function Code
MRBLR4      EQU         SCC4Base+$06      Maximum Receive Buffer Length
CMASK4      EQU         SCC4Base+$34      CRC Constant
CPRES4      EQU         SCC4Base+$38      CRC Preset
MFLR4       EQU         SCC4Base+$46      Maximum Receive Frame Length
RFTHR4      EQU         SCC4Base+$4A      Received Frames Threshold
HMASK4      EQU         SCC4Base+$4E      HDLC Address Mask

*   RISC Timers Parameter RAM

TM_BASE     EQU         TimerBase+$00     RISC Timer Table Base Address
TM_CMD      EQU         TimerBase+$08     RISC Timer Command Register
TM_CNT      EQU         TimerBase+$0C     RISC Timer Counter Register


************* PERFORMANCE TEST INITIALISATION ROUTINE *************

*****************************************************************
* This routine initialises the CP timers and a hardware timer for *
* the performance testing as outlined in section 7.4.10 of the    *
* QUICC User's Manual.                                            *
*                                                                 *
* To evaluate the performance of the QUICC's SCCs, this routine   *
* should be run before starting an application using the SCCs,    *
* SMCs and SPI.                                                   *
*                                                                 *
*****************************************************************

            org         PERFINIT

* Configure general purpose timer 1 to count 16k system clock ticks.
* Timer 1 is cascaded with timer 2 which counts system clocks.
* Do not start timers until the RISC timers have been configured.

            move.w      #0,TGCR           Reset all timers.
            move.w      #$FFFF,TER1       Clear timer event registers.
            move.w      #$FFFF,TER2

            move.w      #$000A,TMR2       Timer 2 counts system clocks
            move.w      #$3FFF,TRR2         up to 16K clocks (a tick).
            move.w      #0,TCN2           Clear timer 2 counter.

            move.w      #0,TMR1           Timer 1 counts 16K clock ticks.
            move.w      #$FFFF,TRR1
            move.w      #0,TCN1           Clear timer 1 counter.

* Configure all 16 RISC timers to count 16k system clock ticks.

            ori.w       #$0F00,RCCR       RISC timer tick = 16K clocks
            move.w      #$04B0,TM_BASE    Start of timer table = $204B0
            move.l      #0,TM_CNT         Clear counter register
            move.w      #$FFFF,RTER       Clear RISC timer event reg.
            move.w      #0,RTMR           Disable all RISC timer interrupts

* One at a time, configure all 16 RISC timers

            move.w      #$0010,D0
            move.l      #$C0000000,D1

TIMEWRITE   move.w      CR,D2             Poll the CR flag and only
            andi.w      #$0001,D2           procede when clear.
            bne         TIMEWRITE

            move.l      D1,TM_CMD         Set up timer command
            move.w      #$0851,CR           and run the command.
```

```
            addi.l      #$00010000,D1     Move to next RISC counter
            subq.w      #1,D0
            bne         TIMEWRITE

LASTWRITE   move.w      CR,D2             Do not procede until final RISC
            andi.w      #$0001,D2           timer command executed.
            bne         LASTWRITE

* Now start the RISC and general purpose timers at the same time.
* Note: interrupts are not disabled while starting the timers so they
*       could start at different times if interrupts are enabled elsewhere
*       in software.

STARTTIME   ori.w       #$8000,RCCR       Start RISC timers
            ori.w       #$0011,TGCR       Start timers 1 and 2.

* Return to QUADS debugger.

            trap        #$F               Exit back
            dc.w        $0063               to ADS debugger



************* PERFORMANCE CHECKING ROUTINE *************

*********************************************************************
* This routine reads the CP timers and the hardware timers and      *
* prints their counter values on the QUADS control terminal. The    *
* user can then compare the hardware and CP timer values. If they   *
* differ by more than two counts, then the CP is running close to   *
* or exceeding its maximum capacity.                                *
*                                                                   *
*  See section 7.4.10 of the QUICC User's Manual for more details.  *
*                                                                   *
*********************************************************************

            org         CHKPERF

            move.l      #0,D0             Clear D0 and D1
            move.l      #0,D1
            move.l      TM_CNT,D0         Read RISC timer tick count
            andi.l      #$0000FFFF,D0     Mask out upper word
            move.w      TCN1,D1           Read timer 1 tick count

            move.l      D0,-(A7)          Push RISC timer value onto stack
            pea         (A7)              Push data stack location
            pea         RISCMSG(PC)       Push string address
            trap        #$F               System call to output
            dc.w        $0028               RISC timer results

            move.l      D1,-(A7)          Push timer 1 value onto stack
            pea         (A7)              Push data stack location
            pea         TIMEMSG(PC)       Push string address
            trap        #$F               System call to output
            dc.w        $0028               timer 1 results

            trap        #$F               Exit back
            dc.w        $0063               to ADS debugger

RISCMSG     DC.B        $1B,'RISC TIMER =    $ ','|10,4Z|',$0A,$0A
TIMEMSG     DC.B        $1B,'GENERAL TIMER = $ ','|10,4Z|',$0A,$0A



************* PERFORMANCE TEST SCC INITIALISATION ROUTINE *************
```

```
*****************************************************************
* This routine initialises four SCCs in HDLC mode and starts    *
* transmission and reception of frames in loopback mode. BDs are *
* used in continuous mode to avoid the need for buffer handling  *
* software. This routine starts the SCCs and requires the user to *
* use the previous performance analysis routine to start the RISC *
* timer and a hardware timer.                                    *
*                                                                *
* SCCs 1,2,3 and 4 are started by this routine. The minimum      *
* configuration is given for each SCC.                           *
*                                                                *
*****************************************************************

            org         SCCINIT

            move.w      #$0740,SDCR       SDMA Configuration Register

* Set up PIO pins

            ori.w       #$00FF,PAPAR      Enable all TXD and RXD signals
            andi.w      #$FF00,PADIR

            ori.l       #$00010038,PBPAR  Enable BRGO1, BRGO2, BRGO3
            andi.l      #$00000038,PBDIR     and BRGO4 signals .
            ori.l       #$00010000,PBDIR

            andi.w      #$F00F,PCSO       SCC CD and CTS inputs always asserted

* SCC clock sources
*    SCC1 clocked by BRG1, SCC2 clocked by BRG2,
*    SCC3 clocked by BRG3, SCC4 clocked by BRG4.
*    Initially set all baud rate generators to approximately 64kbs on a 25Mhz QUICC.

            move.l      #$0001030A,BRGC1  BRG1 = ~64Kbps clock at 25MHz
            move.l      #$0001030A,BRGC2  BRG2 = ~64Kbps clock at 25MHz
            move.l      #$0001030A,BRGC3  BRG3 = ~64Kbps clock at 25MHz
            move.l      #$0001030A,BRGC4  BRG4 = ~64Kbps clock at 25MHz
            move.l      #$1B120900,SICR   Select clock sources

* Configure SCC1 for HDLC

*   SCC1 registers:
            move.l      #$00000040,GSMRL1 SCC in loopback mode and disabled
            move.l      #$00000002,GSMRH1 Send flags between frames
            move.w      #$7E7E,DSR1       HDLC flag pattern
            move.w      #$0000,PSMR1      Clear PSMR
            move.w      #$0000,SCCM1      Mask out interrupts
            move.w      #$FFFF,SCCE1      Clear SCC event register

*   SCC1 parameter RAM:
            move.w      #(SCC1RBD-DPRBASE),RBASE1RX BD Base Address
            move.w      #(SCC1TBD-DPRBASE),TBASE1TX BD Base Address

INITSCC1    move.w      CR,D2             Poll the CR flag and only
            andi.w      #$0001,D2           procede when clear.
            bne         INITSCC1

            move.w      #$0001,CR         Init SCC1 Tx and Rx parameters

WAITSCC1    move.w      CR,D2             Poll the CR flag and only
            andi.w      #$0001,D2           procede when command complete.
            bne         WAITSCC1

            move.b      #$18,RFCR1        SDMA RX Function Code
            move.b      #$19,TFCR1        SDMA TX Function Code
            move.w      #$0020,MRBLR1     Max rx buff length 32 bytes
```

```
            move.l      #$0000F0B8,CMASK1 CCITT CRC-16 mask
            move.l      #$0000FFFF,CPRES1 CCITT CRC-16 preset
            move.w      #$0030,MFLR1      Max rx frame length 48 bytes
            move.w      #$0001,RFTHR1     Rx frames threshold = 1
            move.w      #$0000,HMASK1     Ignore rx frame addresses

* Set up a 32 byte data area ($20, $1F, $1E, .... $02, $01):
            move.l      #SCC1TBUFF,A0
            move.b      #$20,D0
SCC1MSG     move.b      D0,(A0)+
            subq.b      #1,D0
            bne         SCC1MSG

* Set up one tx BD to operate in continuous mode:
            move.l      #SCC1TBD,A0
            move.l      #SCC1TBUFF,(4,A0) Set up BD address field
            move.w      #$0018,(2,A0)     Frame length = 24 bytes
            move.w      #$AE00,(A0)       Set the Ready, Wrap, L, TC and CM bits

* Set up one rx BD to operate in continuous mode:
            move.l      #SCC1RBD,A0
            move.l      #SCC1RBUFF,(4,A0) Set up BD address field
            move.w      #$0000,(2,A0)     Clear length field
            move.w      #$A200,(A0)       Set the Empty, Wrap and CM bits

* Now start the SCC:
            move.l      #$00000070,GSMRL1 SCC in loopback mode and enabled


* Configure SCC2 for HDLC

*   SCC2 registers:
            move.l      #$00000040,GSMRL2 SCC in loopback mode and disabled
            move.l      #$00000002,GSMRH2 Send flags between frames
            move.w      #$7E7E,DSR2       HDLC flag pattern
            move.w      #$0000,PSMR2      Clear PSMR
            move.w      #$0000,SCCM2      Mask out interrupts
            move.w      #$FFFF,SCCE2      Clear SCC event register

*   SCC2 parameter RAM:
            move.w      #(SCC2RBD-DPRBASE),RBASE2RX BD Base Address
            move.w      #(SCC2TBD-DPRBASE),TBASE2TX BD Base Address

INITSCC2    move.w      CR,D2             Poll the CR flag and only
            andi.w      #$0001,D2           procede when clear.
            bne         INITSCC2

            move.w      #$0041,CR         Init SCC2 Tx and Rx parameters

WAITSCC2    move.w      CR,D2             Poll the CR flag and only
            andi.w      #$0001,D2           procede when command complete.
            bne         WAITSCC2

            move.b      #$18,RFCR2        SDMA RX Function Code
            move.b      #$19,TFCR2        SDMA TX Function Code
            move.w      #$0020,MRBLR2     Max rx buff length 32 bytes
            move.l      #$0000F0B8,CMASK2 CCITT CRC-16 mask
            move.l      #$0000FFFF,CPRES2 CCITT CRC-16 preset
            move.w      #$0030,MFLR2      Max rx frame length 48 bytes
            move.w      #$0001,RFTHR2     Rx frames threshold = 1
            move.w      #$0000,HMASK2     Ignore rx frame addresses

* Set up a 32 byte data area ($20, $1F, $1E, .... $02, $01):
            move.l      #SCC2TBUFF,A0
            move.b      #$20,D0
SCC2MSG     move.b      D0,(A0)+
```

```
            subq.b      #1,D0
            bne         SCC2MSG

* Set up one tx BD to operate in continuous mode:
            move.l      #SCC2TBD,A0
            move.l      #SCC2TBUFF,(4,A0) Set up BD address field
            move.w      #$0018,(2,A0)     Frame length = 24 bytes
            move.w      #$AE00,(A0)       Set the Ready, Wrap, L, TC and CM bits

* Set up one rx BD to operate in continuous mode:
            move.l      #SCC2RBD,A0
            move.l      #SCC2RBUFF,(4,A0) Set up BD address field
            move.w      #$0000,(2,A0)     Clear length field
            move.w      #$A200,(A0)       Set the Empty, Wrap and CM bits

* Now start the SCC:
            move.l      #$00000070,GSMRL2 SCC in loopback mode and enabled


* Configure SCC3 for HDLC

*   SCC3 registers:
            move.l      #$00000040,GSMRL3 SCC in loopback mode and disabled
            move.l      #$00000002,GSMRH3 Send flags between frames
            move.w      #$7E7E,DSR3       HDLC flag pattern
            move.w      #$0000,PSMR3      Clear PSMR
            move.w      #$0000,SCCM3      Mask out interrupts
            move.w      #$FFFF,SCCE3      Clear SCC event register

*   SCC3 parameter RAM:
            move.w      #(SCC3RBD-DPRBASE),RBASE3RX BD Base Address
            move.w      #(SCC3TBD-DPRBASE),TBASE3TX BD Base Address

INITSCC3    move.w      CR,D2             Poll the CR flag and only
            andi.w      #$0001,D2           procede when clear.
            bne         INITSCC3

            move.w      #$0081,CR         Init SCC3 Tx and Rx parameters

WAITSCC3    move.w      CR,D2             Poll the CR flag and only
            andi.w      #$0001,D2           procede when command complete.
            bne         WAITSCC3

            move.b      #$18,RFCR3        SDMA RX Function Code
            move.b      #$19,TFCR3        SDMA TX Function Code
            move.w      #$0020,MRBLR3     Max rx buff length 32 bytes
            move.l      #$0000F0B8,CMASK3 CCITT CRC-16 mask
            move.l      #$0000FFFF,CPRES3 CCITT CRC-16 preset
            move.w      #$0030,MFLR3      Max rx frame length 48 bytes
            move.w      #$0001,RFTHR3     Rx frames threshold = 1
            move.w      #$0000,HMASK3     Ignore rx frame addresses

* Set up a 32 byte data area ($20, $1F, $1E, .... $02, $01):
            move.l      #SCC3TBUFF,A0
            move.b      #$20,D0
SCC3MSG     move.b      D0,(A0)+
            subq.b      #1,D0
            bne         SCC3MSG

* Set up one tx BD to operate in continuous mode:
            move.l      #SCC3TBD,A0
            move.l      #SCC3TBUFF,(4,A0) Set up BD address field
            move.w      #$0018,(2,A0)     Frame length = 24 bytes
            move.w      #$AE00,(A0)       Set the Ready, Wrap, L, TC and CM bits

* Set up one rx BD to operate in continuous mode:
```

```
            move.l      #SCC3RBD,A0
            move.l      #SCC3RBUFF,(4,A0) Set up BD address field
            move.w      #$0000,(2,A0)     Clear length field
            move.w      #$A200,(A0)       Set the Empty, Wrap and CM bits

* Now start the SCC:
            move.l      #$00000070,GSMRL3 SCC in loopback mode and enabled


* Configure SCC4 for HDLC

*   SCC4 registers:
            move.l      #$00000040,GSMRL4 SCC in loopback mode and disabled
            move.l      #$00000002,GSMRH4 Send flags between frames
            move.w      #$7E7E,DSR4       HDLC flag pattern
            move.w      #$0000,PSMR4      Clear PSMR
            move.w      #$0000,SCCM4      Mask out interrupts
            move.w      #$FFFF,SCCE4      Clear SCC event register

*   SCC4 parameter RAM:
            move.w      #(SCC4RBD-DPRBASE),RBASE4RX BD Base Address
            move.w      #(SCC4TBD-DPRBASE),TBASE4TX BD Base Address

INITSCC4    move.w      CR,D2             Poll the CR flag and only
            andi.w      #$0001,D2           procede when clear.
            bne         INITSCC4

            move.w      #$00C1,CR         Init SCC4 Tx and Rx parameters

WAITSCC4    move.w      CR,D2             Poll the CR flag and only
            andi.w      #$0001,D2           procede when command complete.
            bne         WAITSCC4

            move.b      #$18,RFCR4        SDMA RX Function Code
            move.b      #$19,TFCR4        SDMA TX Function Code
            move.w      #$0020,MRBLR4     Max rx buff length 32 bytes
            move.l      #$0000F0B8,CMASK4 CCITT CRC-16 mask
            move.l      #$0000FFFF,CPRES4 CCITT CRC-16 preset
            move.w      #$0030,MFLR4      Max rx frame length 48 bytes
            move.w      #$0001,RFTHR4     Rx frames threshold = 1
            move.w      #$0000,HMASK4     Ignore rx frame addresses

* Set up a 32 byte data area ($20, $1F, $1E, .... $02, $01):
            move.l      #SCC4TBUFF,A0
            move.b      #$20,D0
SCC4MSG     move.b      D0,(A0)+
            subq.b      #1,D0
            bne         SCC4MSG

* Set up one tx BD to operate in continuous mode:
            move.l      #SCC4TBD,A0
            move.l      #SCC4TBUFF,(4,A0) Set up BD address field
            move.w      #$0018,(2,A0)     Frame length = 24 bytes
            move.w      #$AE00,(A0)       Set the Ready, Wrap, L, TC and CM bits

* Set up one rx BD to operate in continuous mode:
            move.l      #SCC4RBD,A0
            move.l      #SCC4RBUFF,(4,A0) Set up BD address field
            move.w      #$0000,(2,A0)     Clear length field
            move.w      #$A200,(A0)       Set the Empty, Wrap and CM bits

* Now start the SCC:
            move.l      #$00000070,GSMRL4 SCC in loopback mode and enabled


* Return:
```

```
        trap    #$F             Exit back
        dc.w    $0063             to ADS debugger

END.
```

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

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)
support@freescale.com

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 Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
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