APPLICATION NOTE

AN416
User notes for the SCN68/26562 (NDUSCC) and SC68/26C562 (CDUSCC)

Author: Akber Kazmi

Rev. 1994 Mar

Philips Semiconductors

PHILIPS

User notes for the SCN68/26562 (NDUSCC) and SC68/26C562 (CDUSCC)

AN416

Revised by: A. Kazmi

BISYNC Protocol Questions and Answers

This is a list of some questions and answers for the DUSCC/CDUSCC using BISYNC protocol.

Question:

What is the recommended way for ending a transmit operation?

Answer:

The way to end the transmit operation is to:

– CCR = H‘ 06’	; TEOM after next character
– Put H‘03’ into Tx FIFO	; Send ETX
– Put H ‘FF’ into Tx FIFO	; **Optional** Transmit trailing pad

Now you need to disable TxRDY from interrupting by writing to IER register.

Question:

When is TRSR[7] (TxUnderrun) set at the end of frame?

Answer:

Refer to the Transmitter data path in the data sheet. The status bit TRSR[7] is set when the Tx shift register is empty and no other characters (from the TxFIFO special char. or Sync char.) are waiting to fill it. There can be a one bit time delay due to the Data Encoder after the Tx SR is empty and before the last bit of the character is seen on the TxD pin. The TEOM command causes the FCS to be sent after the next character put into the Tx FIFO is sent. The CRC generation takes place after the Tx SR, so TRSR[7] will be set after the FIFOed character is serialized but before FCS is sent. Another status bit, Frame Complete, TRSR[5] is set when transmission of the FCS begins.

Question:

Are SYN’s in the Tx FIFO excluded from Tx BCC accumulation without using ‘Exclude from CRC’ command (normal mode)?

Answer:

YES.

Question:

Are DLE & SYN in the Tx FIFO excluded from Tx BCC accumulation without using ‘exclude from CRC’ command while in transparent mode?

Answer:

No, not if they are in the FIFO. If DLE is transmitted by TDLE command in the command register, it won’t be accumulated because it won’t go through the FIFO. Any/all characters transmitted through the FIFO in transparent mode will be included in the CRC accumulation. If you don’t want one accumulated, use the ‘exclude from CRC’ command before sending it, or if it is at beginning of frame, use the ‘reset CRC’ command after sending it.

Question:

For BISYNC DMA transfer, is there any way to automatically insert SYNs?

Answer:

One way would be to:

– Program Tx to underrun with SYN s (TPR[7:6] = 11)

– Count down characters to when you want SYN stop the DMA and let the DUSCC Tx underrun

– Start the DMA again after sufficient time to let the DUSCC transmit the SYN

Using this method would preclude using the Tx underrun (TPR[7:6]) to do anything else like underrun with FCS-idle for automatic EOM.

Question:

How could I insert ONLY ONE DLE-SYN in text (in transparent mode)?

Answer:

If you’re not in DMA mode at the point where you want the single DLE-SYN:

– Transmit DLE command (CCR=H’08’)

– Exclude from CRC command (CCR=H’0D’)

– Put SYN character into FIFO

– Proceed with transmitting data characters

If you re using a DMA:

– Program Tx to underrun with SYN s (TPR[7:6] = 11)

– Count down characters (with DMA) to when you want SYN stop the DMA interrupt CPU with DMA let the DUSCC/CDUSCC Tx underrun.

– CPU sets up DMA polls DUSCC/CDUSCC TRSR register and waits for bit 7, Tx empty to get set. The CPU starts DMA again.

This will give at least one DLE-SYN. If CPU is polling TRSR before bit 7 gets set, you will get ONLY ONE DLE-SYN. If the DUSCC/CDUSCC underruns before the CPU is polling for this condition, you may get more than one.

Question:

Does the DUSCC/CDUSCC set the parity bit for ASCII data?

Answer:

The DUSCC/CDUSCC requires that ‘no parity’ be programmed in the CMR1 register and it really doesn’t implement parity. Programming CMR1 [5] = 1 selects that the DUSCC/CDUSCC use its 7-bit odd–parity ASCII look–up table for special character transmission and for reception compares. The CPU must present the DUSCC/CDUSCC with 8-bit data 7 bits plus odd-parity. This requires that the look-up table the CPU uses for the ASCII characters have all 8 bits instead of just 7.

Question:

Does the DUSCC/CDUSCC still receive characters when the BCC check results in a CRC error (after ‘ITB’ received)?

Answer:

Yes.

Question:

What exactly does RPR[7] SYN stripping do while in transparent data mode?

Answer:

A first clarification is that when RPR[7]=0 and you are in BISYNC transparent mode, all odd DLEs are not included in BCC calculation, but are sent through to the Rx FIFO. In BISYNC transparent mode when RPR[7]=1, all odd DLEs will also be stripped so they do not go into the Rx FIFO. Also all occurrences of SYN1 preceded by an odd DLE will be stripped.

In BISYNC normal mode or COP dual SYN mode RPR[7]=1 will enable stripping for all occurrences of SYN1 – SYN2. In single SYN COP mode RPR[7]=1 will enable stripping of all occurrences of SYN1.

Leading SYN patterns (DLE–SYN1 SYN1–SYN2 or SYN1 as appropriate) are always stripped for all modes.

EXAMPLE: Tx Transparent Mode in BISYNC

Assume the part is initialized as follows:

INIT:

```
CMR1=05H      ;COP BISYNC MODE, EBCDIC
CMR2=3FH      ;POLLED/INT MODE, NORMAL, CCIT PRESET 1'S
TTR=3FH       ;38.4K BAUD
RTR=6FH       ;38.4K BAUD
TPR=E3H       ;8 BIT CHAR, UNDERRUN = SYNS, IDLE = SYNS
RPR=83H       ;8 BIT CHAR, STRIP SYN NO FCS TO FIFO OR HUNT
OMR=F7H       ;TXRDY=EMTY, RXRDY=NOT EMPTY, NO RESID CHAR
S1R=66H       ;FIRST SYNC CHAR.=HEX 66
S2R=99H       ;SECOND SYNC CHAR.= HEX 99
CCR=00H       ;RESET TX
CCR=40H       ;RESET RX
CCR=02H       ;ENABLE TX
CCR=42H       ;ENABLE RX
```

Then to start a transparent frame:

```
TXFIFO=55H    ;Put leading pad into TxFIFO, if needed
CCR=05H       ;transmit SOM with PAD command
TRSR[4]=1 ?   ;wait for SOM ACK to be set
CCR=08H       ;transmit DLE before next character command
TXFIFO=02H    ;transmit STX
CCR=01H       ;Reset Tx CRC
```

Now transmit block of transparent data...(can be done with interrupts). Then, to end a transparent frame:

```
CCR=08H       ;transmit DLE before next character command
CCR=006H      ;transmit EOM at end of next character command
TXFIFO=03H    ;transmit ETX
TXFIFO=FFH    ;transmit trailing PAD, if needed
```

Now, do something to keep the transmitter from interrupting until you want to start the next message... (if transmission was interrupt driven)
The receiver will receive: 10 02...XX XX XX...03 (DLE STX ... DATA DATA DATA...ETX) in the receiver FIFO.

SD00394

BISYNC TRANSMISSION

Transmit SOM Sequences for Transparent Mode:

(a) SYN1–SYN2–DLE–STX

– Transmit SOM command	CCR=04H
– Exclude from CRC command	CCR=0DH
– Put DLE into TxFIFO	TxFIFO=10H
– Exclude from CRC command	CCR=0DH
– Put STX into TxFIFO	TxFIFO=02H

–or–

– Transmit SOM command	CCR=04H
– Transmit DLE command	CCR=08H
– Put STX into TxFIFO	TxFIFO=02H
– Reset Tx CRC command	CCR=01H

(b) PAD–SYN1–SYN2–DLE–STX

– Put leading pad into TxFIFO	TxFIFO=55H
– Transmit SOM w/pad command	CCR=05H
– Wait for SOM ACK set	TRSR[4]=1 ?
– Transmit DLE command	CCR=08H
– Put STX into TxFIFO	TxFIFO=02H
– Reset Tx CRC command	CCR=01H

Transmit EOM for Transparent Mode:

DLE–ETX–CRC–CRC–(PAD)

– Transmit DLE command	CCR_=008H
– Transmit EOM command	CCR_=06H
– Put ETX into TxFIFO	TxFIFO=03H
– (optional) Put closing pad into TxFIFO	TxFIFO=FFH

Transmit SOM Sequences for Non–Transparent Mode:

(a) SYN1–SYN2–STX

– Transmit SOM command	CCR=04H
– Exclude from CRC command	CCR=0DH
– Put STX into TxFIFO	TxFIFO=02H

(b) PAD–SYN1–SYN2–STX

– Put leading PAD into TxFIFO	TxFIFO=55H
– Transmit SOM w/PAD command	CCR=05H
– Wait for SOM ACK set	TRSR[4]=1 ?
– Exclude from CRC command	CCR=0DH
– Put STX into TxFIFO	TxFIFO=02H

DLE–SYN Insertion for Transparent Mode:

DLE–SYN1

(a) Not in DMA mode
At point where you want it inserted:

– Transmit DLE command	CCR_=08H
– Exclude from CRC command	CCR_=0DH
– Put SYN1 into TxFIFO	TxFIFO=66H

Proceed on with data transmission (b) In DMA mode

(b) In DMA mode

- Underrun with SYN's programmed at initialization TPR[7:6]=11
- Can use a counter (in DMA or CDUSCC) to time out when you want SYN's
- When counter times out, let the transmitter underrun
- Start transmitting after sufficient time to let the CDUSCC transmit the DLE SYN. You can wait for TRSR[7] = 1 (TxEMPTY) as an indicator of enough time.
- Will get at least one DLE–SYN

Transmit EOM Sequences for Non–Transparent Mode:

ETX–CRC–CRC–(PAD)

(a) Not in DMA mode

– Transmit EOM command	CCR=06H
– Put ETX into TxFIFO	TxFIFO=03H
– (optional) Put closing pad into TxFIFO	TxFIFO=FFH

(b) In DMA mode

- TEOM on zero count or done programmed at initialization TPR[4]=1
- Have ETX as last character in Tx buffer, assert DONEN signal when ETX is written to the CDUSCC

–or–

- If you have programmed to count transmitted characters, program TPR[4] as above,
- ETX should be last character in Tx buffer
- Loading ETX to TxFIFO causes count to go to zero

–or–

- Underrun with FCS–idle programmed at initialization TPR[7:6]=00
- Have ETX as last character in Tx buffer, put into TxFIFO, let Tx underrun.

SD00395

SYN Insertion for Non–Transparent Mode:

SYN1–SYN2

(a) Not in DMA mode

– Put SYN1 into TxFIFO TxFIFO=66H
– Put SYN2 into TxFIFO TxFIFO=99H

(b) In DMA mode

– Underrun with SYN's programmed at initialization, TPR[7:6]=11
– You can use a counter (in DMA or CDUSCC) to time out when you want SYN's
– When counter times out, let the transmitter underrun
– Start transmitting after sufficient time to let the CDUSCC transmit the SYN1–SYN2. You can wait for TRSR[7]=1 (Tx Underrun) as an indicator of enough time.
– You will get at least one SYN1–SYN2

SD00396

BISYNC PROTOCOL WITH DMA

This is an abbreviated flow of the control necessary for BISYNC message transmission and reception under DMA control.

Header Field Transmission Under DMA Control

CPU => Initialization: 1 sec. transmit time-out => Counter
CPU => Initialization: TXU SYN, TEOM on DONE => CDUSCC
CPU => SOH character => Buffer
CPU => Header characters => Buffer
CPU => ETB character => Buffer
CPU => Initialization: TX Buffer address & message length => DMAC
CPU => TXRST => CDUSCC
CPU => ENTX => CDUSCC
CPU => Pad characters => CDUSCC
CPU => Enable TSOM ACK int. => CDUSCC
CPU => TSOM with PAD => CDUSCC
DUSCC =>Int: TSOM ACK => CPU
CPU => Disable TSOM ACK int. => CDUSCC
CPU => EX CRC => CDUSCC
CPU => Enable => DMAC
Buffer => SOH character => CDUSCC
Buffer => Header characters => CDUSCC
Counter =>Int: 1 sec. transmit timeout => CPU
CPU => Disable => DMAC
CPU => Clear TXU status => CDUSCC
CPU => Enable TXU int. => CDUSCC
DUSCC =>DUSCC => Int: TXU => CPU
CPU => Disable TXU int. => CDUSCC
CPU => Enable => DMAC
Buffer => Header characters => CDUSCC

Header Field Terminated Normally:

DMAC => DONE=> CDUSCC
Buffer => ETB character => CDUSCC
DMAC => Int: Count exhausted => CPU
CPU => Disable => DMAC
CPU => Pad character => CDUSCC
CPU => DISTX => CDUSCC
CPU => Initialization: 3 sec. receive timeout => Counter
CPU => Enable SYN detect int. => CDUSCC

Header Field Terminated Prematurely:

CPU => Disable => DMAC
CPU => ENQ character => CDUSCC
CPU => DISTX => CDUSCC
CPU => Initialization: 3 sec. receive timeout => Counter
CPU => Enable SYN detect int. => CDUSCC

Header Field Reception Under DMA Control

CPU => Initialization: RX SYN strip, No FCS to FIFO => CDUSCC
CPU => Initialization: RX Buffer address => DMAC
CPU => Enable => DMAC
CPU => RXRST => CDUSCC
CPU => ENRX => CDUSCC
DUSCC =>Int: SYN detect => CPU
CPU => Clear receive timeout => Counter
DUSCC =>SOH character => Buffer
DUSCC =>Header characters => Buffer
DUSCC =>Int: SYN detect => CPU
CPU => Clear receive timeout => Counter

Header Field Terminated Normally:

DUSCC =>ETB character => Buffer
DUSCC =>DONE => DMAC
DMAC => Int: Frame finished => CPU
DUSCC =>Int: PAD or CRC error => CPU
CPU => DISRX => CDUSCC
CPU => Disable => DMAC

Header Field Terminated Prematurely:

DUSCC =>Int: REOM on ENQ character => CPU
CPU => DISRX => CDUSCC
CPU => Disable => DMAC

Text Field Transmission Under DMA Control

CPU => Initialization: 1 sec. transmit timeout => Counter
CPU => Initialization: TXU SYN, no TEOM on DONE => CDUSCC
CPU => Text characters => Buffer
CPU => Initialization: TX Buffer address & message length => DMAC
CPU => TXRST => CDUSCC
CPU => ENTX => CDUSCC
CPU => Pad characters => CDUSCC

CPU => TSOM with PAD => CDUSCC
CPU => Enable TSOM ACK int. => CDUSCC
DUSCC =>Int: TSOM ACK => CPU
CPU => Disable TSOM ACK int. => CDUSCC
CPU => DLE character => CDUSCC
CPU => STX character => CDUSCC
CPU => RST TX CRC => CDUSCC
CPU => Enable => DMAC
Buffer => Text characters => CDUSCC
Counter =>Int: 1 sec. transmit timeout => CPU
CPU => Disable => DMAC
CPU => Clear TXU status => CDUSCC
CPU => Enable TXU int. => CDUSCC
DUSCC =>Int: TXU => CPU
CPU => Disable TXU int. => CDUSCC
CPU => Enable => DMAC
Buffer => Text characters => CDUSCC
DMAC =>Int: Count exhausted => CPU
CPU => Disable => DMAC
CPU => TDLE => CDUSCC
CPU => TEOM => CDUSCC
CPU => ETX character => CDUSCC
CPU => Pad character => CDUSCC
CPU => DISTX => CDUSCC
CPU => Initialization: 3 sec. receive timeout => Counter
CPU => Enable SYN detect int. => CDUSCC

Text Field Reception Under DMA Control

RX SYN strip, No FCS to FIFO => CDUSCC
CPU => Initialization: RX Buffer address => DMAC
CPU => Enable => DMAC
CPU => RXRST => CDUSCC
CPU => ENRX => CDUSCC
DUSCC =>Int: SYN detect => CPU
CPU => Clear receive timeout => Counter
DUSCC =>DLE character => Buffer
DUSCC =>STX character => Buffer
DUSCC =>Text characters => Buffer
DUSCC =>Int: SYN detect => CPU
CPU => Clear receive timeout => Counter
DUSCC =>ETX character => Buffer
DUSCC =>DONE => DMAC
DMAC => Int: Frame finished => CPU
DUSCC =>Int: PAD or CRC error => CPU
CPU => DISRX => CDUSCC
CPU => Disable => DMAC

Control Field Transmission

CPU => Initialization: 1 sec. transmit timeout => Counter
CPU => Initialization: TXU SYN, no TEOM on DONE => CDUSCC
CPU => TXRST => CDUSCC
CPU => ENTX => CDUSCC
CPU => Pad characters => CDUSCC
CPU => TSOM with PAD => CDUSCC
CPU => Enable TSOM ACK int. => CDUSCC
DUSCC =>Int: TSOM ACK => CPU
CPU => Disable TSOM ACK int. => CDUSCC
CPU => Control characters => CDUSCC
Counter =>Int: 1 sec. transmit timeout => CPU
CPU => Clear TXU status => CDUSCC
CPU => Enable TXU int. => CDUSCC
DUSCC =>Int: TXU => CPU
CPU => Disable TXU int. => CDUSCC
CPU => Control characters => CDUSCC
CPU => Pad characters => CDUSCC
CPU => DISTX => CDUSCC
CPU => Initialization: 3 sec. receive timeout => Counter
CPU => Enable SYN detect int. => CDUSCC

Control Field Reception

CPU => Initialization: RX SYN strip, No FCS to FIFO => CDUSCC
CPU => RXRST => CDUSCC
CPU => ENRX => CDUSCC
DUSCC =>Int: SYN detect => CPU
CPU => Clear receive timeout => Counter
DUSCC =>Control characters => Buffer
DUSCC =>Int: REOM on control character terminator => CPU
DUSCC =>Int: PAD error => CPU
CPU => DISRX => CDUSCC;

```
;INTERRUPT DRIVEN, TRANSPARENT MODE BISYNC EXAMPLE
;
;THIS IS PROGRAM BISYNC_INT. IT RUNS WITH AN APPLICATIONS 68K
;BOARD, THE DUSCC BENCH BOARD, AND CH A EXT.  CONNECTED TO CH B,
;
;D. IBARRA JAN. 1988
                BEGIN
OMRA            EQU         $74017          ;OUTPUT & MISC. A & B
OMRB            EQU         $74057
CMR1A           EQU         $74001          ;CHAN MODE REGS
CMR1B           EQU         $74041
CMR2A           EQU         $74003
CMR2B           EQU         $74043
S1RA            EQU         $74005          ;SYN1
S1RB            EQU         $74045
S2RA            EQU         $74007          ;SYN2
S2RB            EQU         $74047
TPRA            EQU         $74009
TTRA            EQU         $740OB          ;TXA PARAMETER
TPRB            EQU         $74049          ;TXA TIMING
TTRB            EQU         $7404B
RPRA            EQU         $740OD
RTRA            EQU         $7400F
RPRB            EQU         $7404D          ;RXB PARAMETER
RTRB            EQU         $7404F          ;RXB TIMING
GSR             EQU         $74037          ;GENERAL STATUS REG
CCRA            EQU         $7401F          ;CHAN COMMAND REG A & B
CCRB            EQU         $7405F
TXFIFA          EQU         $74021          ;TXA FIFO
TXFIFB          EQU         $74061
RXFIFA          EQU         $74029
RXFIFB          EQU         $74069          ;RXB FIFO
PCRA            EQU         $7401D
PCRB            EQU         $7405D
TRSRA           EQU         $74033          ;TX/RX STATUS REG
TRSRB           EQU         $74073
RSRA            EQU         $74031
RSRB            EQU         $74071
IVR             EQU         $7403D
IVRM            EQU         $7407D
ICR             EQU         $7403F
IERA            EQU         $74039
IERB            EQU         $74079
;
START:          BSR         INIT            ;INITIALIZE DUSCC
                BSR         SETINT          ;SET UP INTERRUPTS
                LEA         TXBUF,A1        ;TX BUFFER POINTER
                LEA         RXBUF,A2        ;RX BUFFER POINTER
                MOVE.B      #$C3,CCRA       ;SET NRZ MODE FOR DPLL
                MOVE.B      #$C3,CCRB       ;SET NRZ MODE FOR DPLL
                MOVE.B      #$C0,CCRB       ;ENTER SEARCH MODE (DPLL)
                BSR         STFRM           ;TRANSMIT START OF TRNSP. FRAME
                MOVE.B      #$40,IERA       ;ENABLE TX A INT.
                MOVE.B      #$10,IERB       ;ENABLE RX B INT.
;
WT              STOP        #$2000          ;SUPERVISOR MODE,ANY INT,NO TRAP
                JMP         WT              ;
```

SD00397

```
;
;————————INITIALIZATION ROUTINES——————————
;
;INIT:      MOVE.B    #5,CMR1A       ;COP BISYNC MODE, EBCDIC
            MOVE.B    #5,CMR1B       ;COP BISYNC MODE, EBCDIC
            MOVE.B    #$3F,CMR2A     ;POLLED/INT MODE, NORMAL, CCIT PRESET 1'S
            MOVE.B    #$3F,CMR2B     ;POLLED/INT MODE, NORMAL, CCIT PRESET 1'S
            MOVE.B    #$3F,TTRA      ;38.4K BAUD
            MOVE.B    #$3F,TTRB      ;38.4K BAUD
            MOVE.B    #$6F,RTRA      ;38.4K BAUD
            MOVE.B    #$6F,RTRB      ;38.4K BAUD
            MOVE.B    #$E3,TPRA      ;TX=8 BIT/CHAR, UNDERRUN=SYNS, IDLE=SYNS
            MOVE.B    #$E3,TPRB      ;TX=8 BIT/CHAR, UNDERRUN=SYNS, IDLE=SYNS
            MOVE.B    #$83,RPRA      ;RX=8 BIT/CHAR,STRIP SYN
            MOVE.B    #$83,RPRB      ;RX=8 BIT/CHAR,STRIP SYN
            MOVE.B    #$E7,OMRA      ;TXRDY=NOT FULL, RXRDY=NOT EMPTY, NO RESID CHAR
            MOVE.B    #$E7,OMRB      ;TXRDY=NOT FULL, RXRDY=NOT EMPTY, NO RESID CHAR
            MOVE.B    #$66,S1RA      ;FIRST SYNC CHAR.=HEX 66
            MOVE.B    #$66,S1RB      ;FIRST SYNC CHAR.=HEX 66
            MOVE.B    #$99,S2RA      ;SECOND SYNC CHAR.=HEX 99
            MOVE.B    #$99,S2RB      ;SECOND SYNC CHAR.=HEX 99
            MOVE.B    #0,CCRA        ;RESET TX A
            MOVE.B    #0,CCRB        ;RESET TX B
            MOVE.B    #$40,CCRA      ;RESET RX A
            MOVE.B    #$40,CCRB      ;RESET RX B
            MOVE.B    #2,CCRA        ;ENABLE TX A
            MOVE.B    #2,CCRB        ;ENABLE TX B
            MOVE.B    #$42,CCRA      ;ENABLE RX A
            MOVE.B    #$42,CCRB      ;ENABLE RX B
            RTS
;
SETINT:     MOVEA.L   $110,A6        ;GET ADDRESS AT VECTOR 68
            MOVE.L    #RXB,2[A6]     ;RXB INT. ROUTINE ADD. TO JUMP INST.
            MOVEA.L   $104,A6        ;GET ADDRESS AT VECTOR 65
            MOVE.L    #TDBUF,2[A6]   ;TDBUF INT. ROUTINE ADD. TO JUMP INST.
            MOVE.B    #64,IVR        ;INT. VECTOR V64 INTO DUSCC
            MOVE.B    #$C7,ICR       ;INTRLVD, B PRTY, A&B ENBL, VECT. INC. STATUS
            RTS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; TRANSMIT ROUTINES ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;SEQ. TO START TRANSPARENT DATA FRAME
;
STFRM:      MOVE.B    #$55,TXFIFA    ;PUT LEADING PAD INTO TXFIFA
            MOVE.B    #$05,CCRA      ;TRANSMIT SOM WITH PAD
WTSOM       MOVE.B    TRSRA,D5       ;READ STATUS
            BTST      #4,D5          ;IS SOM ACK SET?
            BEQ       WTSOM          ;IF NOT, WAIT 'TILL IT IS
            MOVE.B    #$08,CCRA      ;TRANSMIT DLE BEFORE NEXT CHAR.
            MOVE.B    #$02,TXFIFA    ;TRANSMIT STX
            MOVE.B    #$01,CCRA      ;RESET TX CRC
            RTS
;
;TRANSMIT FROM DATA BUFFER, INTERRUPT ROUTINE
;
TDBUF:      MOVE.B    [A1]+,TXFIFA   ;SEND NEXT CHAR.
            CMPA.L    #RXBUF,A1      ;AT END OF CHAR. BUFFER?
            BEQ       ETFRM          ;IF LAST CHAR, END FRAME
            RTE
```

SD00398

```
;
;SEQ. TO END TRANSPARENT DATA FRAME
;
ETFRM:      BSR WTXRDY                  ;WAIT FOR TXRDY
            MOVE.B      #$08,CCRA       ;TRANSMIT DLE BEFORE NEXT CHARACTER
            MOVE.B      #$06,CCRA       ;TRANSMIT EOM AT END OF NEXT CHARACTER
            MOVE.B      #$03,TXFIFA     ;TRANSMIT ETX
;
;————————CAN'T DO YET, BECAUSE OF ANOMALY————————————————————
;           BSR         WTXRDY          ;WAIT FOR TXRDY                                ;
;           MOVE.B      #$FF,TXFIFA     ;TRANSMIT TRAILING PAD                         ;
;———————————————————————————————————————————————————————————;
;THIS WILL KEEP TX FROM INTERRUPTING UNTIL WANT TO START NEXT MESSAGE
;
RSTRDY:     MOVE.B      #$85,ICR        ;TURN OFF CH A INTERRUPT
;
;————————CAN'T DO YET, BECAUSE OF ANOMALY————————————————————
            RSTRDY:     BSR WTXRDY      ;WAIT FOR TXRDY                                ;
;           MOVE.B      #$02,GSR        ;RESET TXRDY BIT                               ;
;———————————————————————————————————————————————————————————
            RTE
;
;THIS SUBROUTINE WAITS FOR TXRDY
;
WTXRDY:     MOVE.B      GSR,D0          ;READ GSR TO D0
            BTST        #1,D0           ;IS TXRDY A SET ?
            BEQ         WTXRDY          ;IF NOT, WAIT TILL IT IS
            RTS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; RX READY INTERRUPT ROUTINE ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
RXB:        MOVE.B      RSRB,D3         ;READ RECEIVER STATUS REG.
            MOVE.B      TRSRB,D4        ;READ TX/RX STATUS REG.
            MOVE.B      TRSRA,D2        ;TEMP. READ OF TX STATUS
            BTST        #5,D3           ;OVERRUN ERROR ?
            BNE         RXERR           ;IF YES, GO TO ERROR HANDLER
            BTST        #7,D3           ;EOM DETECT ?
            BNE         RXEND           ;IF YES, GO TO RECEIVED END
            MOVE.B      RXFIFB, [A2]+   ;READ CHAR. TO BUFFER
            RTE
;
RXEND:      BTST        #1,D3           ;CRC ERROR ?
            BNE         RXERR           ;IF YES, GO TO ERROR HANDLER
            MOVE.B      RXFIFB, [A2]+   ;READ CHAR. TO BUFFER
            TRAP        #15             ;STOP AND DISPLAY STATUS
;
RXERR:      TRAP        #15             ;CRC ERROR ?
            BNE         RXERR           ;IF YES, GO TO ERROR HANDLER
            MOVE.B      RXFIFB, [A2]+   ;READ CHAR. TO BUFFER
            TRAP        #15             ;STOP AND DISPLAY STATUS
;
RXERR:      TRAP        #15             ;STOP AND DISPLAY STATUS
;
;—————————————————— DATA BUFFERS ——————————————————
;
TDBUF:      DC.B        0,1,2,3,4,5,6,7,8,9,10
RXBUF       DS.B        15
;
            END START
```

SD00399

HANDLING DDCMP IN THE DUSCC/CDUSCC RECEIVER

There are two operations that require special handling:

1. The text field character count is contained in the header field and must be loaded in the counter/timer before the text field begins.
2. In non-BISYNC COP, the CRC error status bit (RSR[1]) is updated every time a character is loaded in the receive FIFO. The CRC accumulates to the proper value only during the last byte of the CRC, so all other characters are appended with a CRC error. RSR[1] does not clear after each character so it must be reset by the CPU after the first byte of the CRC to accurately reflect the CRC status of the second byte of CRC.

The following can be done to perform these functions:

1. The counter timer counts the number of characters in the header field. The second and third bytes of this field contain the text field character count. When this value is received it is loaded into the counter timer preset register so that when the counter reaches zero at the end of the header field it will be loaded with the text field character count.
2. It is important to know when the first byte of the CRC is at the top of the FIFO, because RSR[1] must be reset by the CPU before the second byte of the CRC is at the top of the FIFO. Therefore, load the counter timer with a count which is one less than the length of the frame so an interrupt will occur when the first byte of CRC is at the top of the FIFO.

The following sequence illustrates how a typical DDCMP frame can be handled in the receiver:

– Initialize for DDCMP protocol.

– Set Receive characters as C/T clock source, CTCR[2:O] = 1 1 0.

– Load the C/T with a count which is one less than the length of header field so that Char. Count Complete indicator, RSR[7], will be set when the first byte of CRC is at the top of the FIFO.

– Enable receiver

– Start C/T

– Start receiving header field characters. As soon as text field character count is received, load it into the C/T (CTPRH/L registers). This will not affect the current count in progress. It will be loaded by C/T when current count is complete.

– Continue receiving header field characters, look for RSR[1] to be set (Char. Count Complete indicator) before reading each character from FIFO.

– When RSR[7l is set, CRC1 byte is at the top of the FIFO. Before reading CRC1 from FIFO, clear RSR[1] (CRC error) and RSR[7]. Read CRC1 from FIFO.

– CRC2 is now at the top of FIFO. RSR[1] will now correctly indicate whether the header has had a CRC error.

– Read CRC2 from FIFO

– Char. Count Complete, RSR[7], can generate an interrupt by setting IER[3] (enable interrupt for RSR[7:6]), and setting the master interrupt enable in ICR.

BOP PROTOCOL QUESTIONS AND ANSWERS

Question:

Using the DUSCC/CDUSCC in BOP mode, you would like the DUSCC/CDUSCC transmitter to negate RTS when done with a frame, but if you have back-to-back frames, you don’t want it to negate until after the last frame.

Answer:

Should have no problem with this. During the initialization program TPR[3] = 1, the transmitter controls RTS. When you first enable the transmitter, you need to manually assert RTS by writing to OMR. Disable the transmitter after loading the last character into the FIFO, and RTS will negate five bit times after transmission of the last bit of the closing FLAG. If the transmitter is re-enabled for transmission of a subsequent frame before the five bit time delay has elapsed, RTS will not negate.

Question:

You want to transmit a break in between transmission of data characters; also you want data character, break, mark, and then data again to go out on the line. What is the best way to do this? Can data characters be left in the TxFIFO while you transmit the break?

Answer:

Data cannot be left in the TxFIFO when you give the Rx BREAK command. Invoking the Transmit BREAK command will cause the transmitter FIFO to be flushed. A data character in the Tx Shift Register will still be transmitted after you give the Tx BREAK command before the BREAK is transmitted.

The transmitter looks at the state of Tx enabled or disabled at the character boundary when it is done sending the break. If it is disabled, it goes to mark; if it is enabled, it will send another break. The TxFIFO is actually flushed when the Tx BREAK ACK is set to indicate the BREAK has started transmission. So, you need to know when the BREAK is done before having the Tx re-enabled. A good way to know when it is done is to ask for a second BREAK, and when you get the BREAK ACK for it, we know the first one has gone out. Then, we need to do a Tx RESET to kill it, since a quick disable/enable will be seen as enable at the second end of BREAK boundary. Tx RESET will immediately bring the Tx output pin high and the second break will be ignored. So, the recommended sequence is to:

– Wait for GSR[7]=1, TxRDY (with OMR[4] = 1, TxRDY = FIFOEMTY)

– Issue Tx BREAK command, CCR=H’07’

– Wait for TRSR[4]=1, Tx BREAK ACK set

– Write ‘1’ to TRSR[4]

– Issue Tx BREAK command, CCR=H‘07’

– Wait for TRSR[4]=1

– Issue Tx RESET command, CCR=H‘00’

– Issue Tx ENABLE command, CCR=H‘02’

– Put data into Tx FIFO

Output of Tx will look like:

– DATA – BREAK – MARK – DATA –

Question:
Can the sequence ‘TxABORT–TSOM–Data to FIFO’ be done in this sequence with no problems? What is the best way to transmit ABORT and continue on with next frame?

Answer:
Invoking the Tx ABORT command will flush any characters in the TxFIFO, as mentioned above. This is done right before the Tx ABORT is sent out, so you don’t want to do the exact sequence in the question. The TSOM command can be invoked as soon as TRSR[4], ABORT ACK, is set. New data characters can be loaded into the Tx FIFO one bit time after TRSR[4] is set. This one bit time is needed because the internal command to clear the Tx FIFO is asserted when the ACK is set, and it lasts one bit time. So, the sequence should be:

– Tx ABORT command

– Wait for TRSR[4]=1, ABORT ACK set

– TSOM command

– Delay, if needed, to have one bit time delay

– New data char. into Tx FIFO

Question:
I’m not getting my last character transmitted in my interrupt routine after TEOM is set.

Answer:
If you had residual character length set at the default value (OMR[7:5]–000) of 1 bit. So, the Tx sent out 1 bit of the last character. The solution is to program the residual character length to be same as the Tx character length (OMR[7:5] – 111).

Question:
At slower speeds, I see a time difference in getting EOM and Flag detect interrupts. Aren’t these caused by the same event? Why the time difference?

Answer:
In BOP mode, receiving the closing Flag does indicate the end of frame. When the receiver detects the closing flag, it uses the 16 bits it received prior as the CRC, and appends EOM detect indicator to the last character in the FIFO (this is usually the last character in the information field, but if CRC is sent to FIFO, this will be appended to the last byte of CRC). Now, as far as the RSR bits are concerned, the Flag detect bit will always be set first. This is because the Flag detect is set as soon as the Flag is received. The EOM detect bit is set when the last character reaches the top of the FIFO, which always happens at least two bit times after Flag detect is set (longer if CRC is sent to FIFO or FIFO has previous characters still in it).

Question:
Customer is using SDLC protocol, they want to send an abort sequence followed by a 2 byte preframe before the normal frame. How is the preframe sent?

Answer:
To send the preframe they need to do the following after sending their abort sequence:

– Load the 2 characters they want to use for the preframe into the TxFIFO

– Transmit start of message with pad, CCRA[7,0] – 00XX0101

– After start of message has been sent, load TxFIFO with the message. You can check for this by polling TRSR[4] until it sets.

Question:
What is a way to get the TxD output continuously ‘0’ for the call sequence?

Answer:
There are two ways to do this, one uses only software and the other needs external hardware. The software implementation is to put all zero characters into the TxFIFO and use the transmit start of message with PAD command (TSOMP). Be sure to keep the TxFIFO full of zero characters for as long as the continuous zero is needed. The other way is to use the GPO output on the DUSCC to control whether TxD or ‘0’ is output on the data line. The hardware would implement TxD ANDed with GPO to get TxD. When GPO is negated (high) the Tx Data will go through, when GPO is asserted (low) the Tx Data line will be continuously low.

Question:
How can the transmitter be synchronized with an external sync. signal to implement transmitter byte timing?

Answer:
There is no way internal to the DUSCC to synchronize the transmitter with an external sync. signal. The ‘External Sync Input’ cannot be used for the transmitter. The transmitter byte timing synchronization would need to be done external to the chip. This would require a fair amount of external hardware to implement (estimate at least 3 packages). The transmitter byte timing requirement is fill option in the X.21 spec., some countries have standards which use this and others don’t.

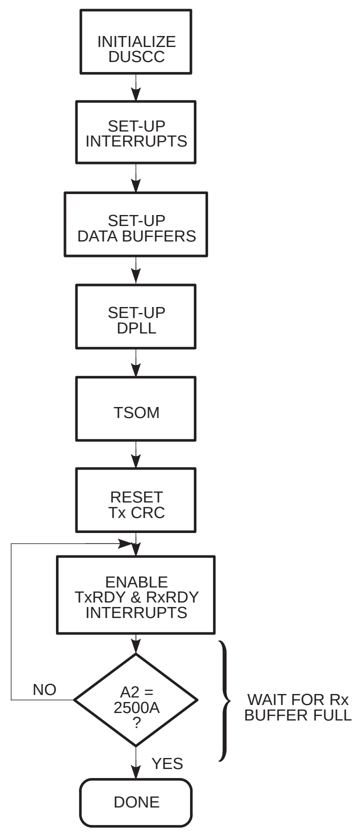
```
; INTERRUPT DRIVEN, BOP PROTOCOL EXAMPLE
; THIS IS PROGRAM BOP_INT.
; CHA SENDS AND RECEIVES CHARACTERS IN BOP
; USING INTERRUPT ROUTINES.
; CHA RX AND TX EXTERNALLY TIED
;
; D. IBARRA AUG., 1987, UPDATED DEC. 1988
;
                    BEGIN

OMRA      EQU       $74017        ;OUTPUT & MISC. A & B
OMRB      EQU       $74057
CMR1A     EQU       $74001        ;CHAN MODE REGS
CMR1B     EQU       $74041
CMR2A     EQU       $74003
CMR2B     EQU       $74043
TPRA      EQU       $74009        ;TXA PARAMETER
TTRA      EQU       $7400B        ;TXA TIMING
TPRB      EQU       $74049
TTRB      EQU       $7404B
RPRA      EQU       $7400D
RTRA      EQU       $7400F
RPRB      EQU       $7404D        ;RXB PARAMETER
RTRB      EQU       $7404F        ;RXB TIMING
GSR       EQU       $74037        ;GENERAL STATUS REG
CCRA      EQU       $7401F        ;CHAN COMMAND REG A & B
CCRB      EQU       $7405F
S1RA      EQU       $74005        ;SECONDARY ADDRESS REGISTER
TXFIFA    EQU       $74021        ;TXA FIFO
TXFIFB    EQU       $74061
RXFIFA    EQU       $74029
RXFIFB    EQU       $74069        ;RXB FIFO
PCRA      EQU       $7401D
PCRB      EQU       $7405D
RSRA      EQU       $74031
TRSRA     EQU       $74033        ;TX/RX STATUS REG
IERA      EQU       $74039
IVR       EQU       $7403D
ICR       EQU       $7403F
;
START:    BSR       INIT          ;INITIALIZE PART
          BSR       SETINT        ;SET UP INTERRUPTS
          LEA       TXBUF,A1      ;TX BUFFER POINTER
          LEA       RXBUF,A2      ;RX BUFFER POINTER
          MOVE.B    #$C2,CCRA     ;MANCHESTER
          MOVE.B    #$C0,CCRA     ;DPLL ENTER SEARCH MODE
          MOVE.B    #$04,CCRA     ;TSOM
          MOVE.B    #$01,CCRA     ;RESET TX CRC
          MOVE.B    #$50,IERA     ;ENABLE TXRDY AND RXRDY INT.
;
WTDN:     STOP      #$2000        ;WAIT FOR INTERRUPT ROUTINES TO END PROGRAM
          JMP       WTDN          ;
```

SD00400

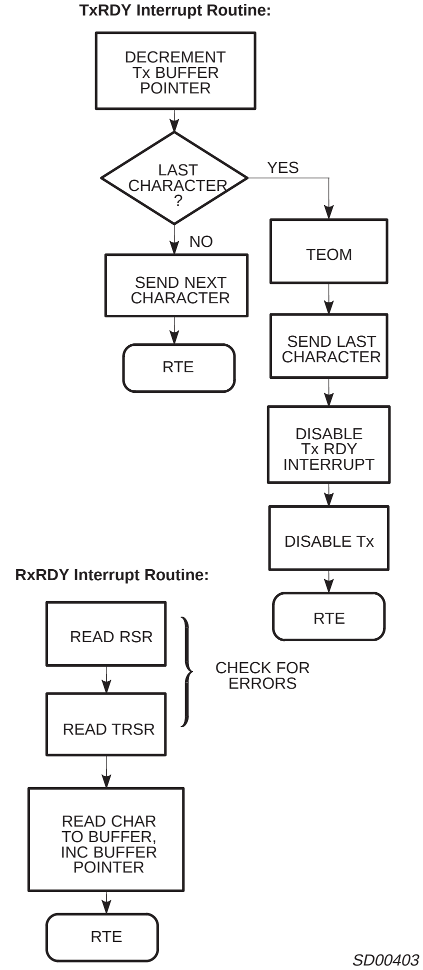
```
;————————————————————————————————————————————————————————————————————————————
;——————————————————————————————SUBROUTINES————————————————————————————————————
;————————————————————————————————————————————————————————————————————————————
INIT:      MOVE.B    #$00,CCRA      ;RESET TX
           MOVE.B    #$40,CCRA      ;RESET RX
           MOVE.B    #$00,CMR1A     ;8-BIT ADD., BOP PRIMARY
           MOVE.B    #$3F,CMR2A     ;NORMAL, POLLED/INT.
           MOVE.B    #$06,PCRA      ;TXC ON TRXC
           MOVE.B    #$23,TPRA      ;UNDRN=FCS=FLAG–IDLE,IDLE=FLAGS, 8 BITS
           MOVE.B    #$23,RPRA      ;OVRN=CONTINUE FRAME, 8 BITS
           MOVE.B    #$3D,TTRA      ;TXC=BRG 9600 BAUD
           MOVE.B    #$6D,RTRA      ;RXC=DPLL, 9600 BAUD FROM BRG
           MOVE.B    #$E0,OMRA      ;TXRDY=NOT FULL, RXRDY=NOT EMPTY
           MOVE.B    #$00,CCRA      ;RESET TX
           MOVE.B    #$40,CCRA      ;RESET RX
           MOVE.B    #$02,CCRA      ;ENABLE TX
           MOVE.B    #$42,CCRA      ;ENABLE RX
           RTS
;
SETINT:    MOVE.L    #RXINT,$100    ;RX INT. ROUTINE ADD. TO VECTOR 64
           MOVE.L    #TXINT,$104    ;TX INT. ROUTINE ADD. TO VECTOR 65
           MOVE.B    #64,IVR        ;INT. VECTOR 64 INTO DUSCC
           MOVE.B    #$06,ICR       ;CHA ENABLE, VECTOR INC. STATUS
           RTS
           MOVE.B    #$C7,ICR       ;INTRLVD, B PRTY, A&B ENBL, VECT. INC. STATUS
           RTS
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;INTERRUPT ROUTINES;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
TXINT:     SUBA.L    #1,A1          ;DECREMENT TX BUFFER POINTER
           CMPA.L    #RXBUF,A1      ;LAST CHAR ?
           BNE       SEND           ;IF NOT, SEND NEXT CHAR
           MOVE.B    #$06,CCRA      ;TEOM
           MOVE.B    A1,TXFIFA      ;SEND LAST CHAR.
           MOVE.B    #$10,IERA      ;INT. ON RXRDY ONLY
           MOVE.B    #$03,CCRA      ;DISABLE TX
           RTE
SEND:      MOVE.B    [A1],TXFIFA    ;SEND NEXT CHAR.
           RTE
;
RXINT:     MOVE.B    RSRA,D3        ;RECEIVER STATUS TO D3
           MOVE.B    TRSRA,D4       ;TX/RX STATUS TO D4
           BTST      #7,D3          ;EOM DETECT?
           BNE       RXEND          ;IF YES, GO TO RECEIVED END
           MOVE.B    RXFIFA,[A2]+   ;READ CHAR. TO BUFFER
           RTE
;
RXEND:     MOVE.B    RXFIFA,,[A2]+  ;READ CHARACTER TO BUFFER
           TRAP      #15            ;STOP AND DISPLAY STATUS
;
;————————————————————————————————————DATA BUFFERS—————————————————————————————————
TXBUF      DC.B      0,1,2,3,4,5,6,7,8,9,10
RXBUF      DS.B      15
;
           END       START
```

SD00400



DUSCC_BOP_INT, Main Program Flow Chart

SD00402



DUSCC_BOP INT TxRDY and RxRDY Flow Chart

```
; INTERRUPT DRIVEN, ASYNCHRONOUS PROTOCOL EXAMPLE
; THIS IS PROGRAM ASYNC INT.  IT RUNS WITH AN APPLICATIONS 68K
; BOARD, THE DUSCC BENCH BOARD, AND CH A EXT.  CONNECTED TO CH. B
;
; D. IBARRA NOV. 1988
;
                        BEGIN

OMRA      EQU           $74017          ;OUTPUT & MISC. A & B
OMRB      EQU           $74057
CMR1A     EQU           $74001          ;CHAN MODE REGS
CMR1B     EQU           $74041
CMR2A     EQU           $74003
CMR2B     EQU           $74043
S1RA      EQU           $74005          ;SYN1
S1RB      EQU           $74045
S2RA      EQU           $74007          ;SYN2
S2RB      EQU           $74047
TPRA      EQU           $74009          ;TXA PARAMETER
TTRA      EQU           $7400B          ;TXA TIMING
TPRB      EQU           $74049
TTRB      EQU           $7404B
RPRA      EQU           $7400D
RTRA      EQU           $7400F
RPRB      EQU           $7404D          ;RXB PARAMETER
RTRB      EQU           $7404F          ;RXB TIMING
GSR       EQU           $74037          ;GENERAL STATUS REG
CCRA      EQU           $7401F          ;CHAN COMMAND REG A & B
CCRB      EQU           $7405F
TXFIFA    EQU           $74021          ;TXA FIFO
TXFIFB    EQU           $74061
RXFIFA    EQU           $74029
RXFIFB    EQU           $74069          ;RXB FIFO
PCRA      EQU           $7401D
PCRB      EQU           $7405D
TRSRA     EQU           $74033          ;TX/RX STATUS REG
TRSRB     EQU           $74073
RSRA      EQU           $74031
RSRB      EQU           $74071
IVR       EQU           $7403D
IVRM      EQU           $7407D
ICR       EQU           $7403F
IERA      EQU           $74039
IERB      EQU           $74079
;
START:    BSR           INIT            ;INITIALIZE DUSCC
          BSR           SETINT          ;SET UP INTERRUPTS
          LEA           TXBUF,A1        ;TX BUFFER POINTER
          LEA           RXBUF,A2        ;RX BUFFER POINTER
          MOVE.B        #$40,IERA       ;ENABLE TX A INT.
          MOVE.B        #$10,IERB       ;ENABLE RX B INT.
;
WT:       STOP          #$2000          ;SUPERVISOR MODE, ANY INT, NO TRAP
          JMP           WT              ;
;
;———————————————————INITIALIZATION ROUTINES———————————————————————
;
INIT:     MOVE.B #7,CMR1A               ;ASYNC, NO PARITY
          MOVE.B #7,CMR1B               ;ASYNC, NO PARITY
          MOVE.B #$38,CMR2A             ;POLLED/INT MODE, NORMAL
          MOVE.B #$38,CMR2B             ;POLLED/INT MODE, NORMAL
          MOVE.B #$3F,TTRA              ;38.4K BAUD
          MOVE.B #$3F,TTRB              ;38.4K BAUD
          MOVE.B #$2F,RTRA              ;38.4K BAUD
```

SD00404

```
                MOVE.B    #$2F,RTRB        ;38.4K BAUD
                MOVE.B    #$F3,TPRA        ;TX=8 BIT/CHAR,2 STOP BITS
                MOVE.B    #$F3,TPRB        ;TX=8 BIT/CHAR,2 STOP BITS
                MOVE.B    #$03,RPRA        ;RX=8 BIT/CHAR
                MOVE.B    #$03,RPRB        ;RX=8 BIT/CHAR
                MOVE.B    #$00,OMRA        ;TXRDY=NOT FULL, RXRDY=NOT EMPTY
                MOVE.B    #$00,OMRB        ;TXRDY=NOT FULL, RXRDY=NOT EMPTY
                MOVE.B    #0,CCRA          ;RESET TX A
                MOVE.B    #0,CCRB          ;RESET TX B
                MOVE.B    #2,CCRA          ;ENABLE TX A
                MOVE.B    #2,CCRB          ;ENABLE TX B
                RTS
;
SETINT:         MOVEA.L   $110,A6          ;GET ADDRESS AT VECTOR 68
                MOVE.L    #RXB,2[A6]       ;RXB INT. ROUTINE ADD. TO JUMP INST.
                MOVEA.L   $104.A6          ;GET ADDRESS AT VECTOR 65
                MOVE.L    #TDBUF,2[A6]     ;TDBUFF INT. ROUTINE ADD. TO JUMP INST.
                MOVE.B    #64,IVR          ;INT. VECTOR V64 INTO DUSCC
                MOVE.B    #$C7,ICR         ;INTRLVD, B PRTY, A&B ENBL, VECT. INC. STATUS
                RTS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;TX READY INTERRUPT ROUTINE;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
TDBUF:          MOVE.B    [A1]+,TXFIFA     ;SEND NEXT CHAR.
                CMPA.L    #RXBUF,A1        ;AT END OF CHAR. BUFFER?
                BEQ       DISINT           ;IF LAST CHAR, DISABLE INTERRUPT
                RTE

;
;THIS WILL KEEP TX FROM INTERRUPTING UNTIL WANT TO START NEXT MESSAGE
;
DISINIT:        MOVE.B    #$85,ICR         ;TURN OFF CH A INTERRUPT
                RTE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;RX READY INTERRUPT ROUTINE;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
RXB:            MOVE.B    RSRB,D3          ;READ RECEIVER STATUS REG.
                BTST      #5,D3            ;OVERRUN ERROR?
                BNE       RXERR            ;IF YES, GO TO ERROR HANDLER
                MOVE.B    RXFIFB,[A2]+     ;READ CHAR. TO BUFFER
                CMPA.L    #DONE,A2         ;AT END OF CHAR BUFFER?
                BEQ       STOP             ;IF LAST CHAR, STOP
                RTE
;
RXERR:          TRAP      #15              ;STOP AND DISPLAY STATUS
STOP:           TRAP      #15              ;STOP AND DISPLAY STATUS
;
;——————————————————————————DATA BUFFERS——————————————————————————
;
TXBUF           DC.B      0,1,2,3,4,5,6,7,8,9,10
RXBUF           DS.B      11
;
DONE:           END       START
```

SD00405

DESIGN CAUTIONS

– The system clock must be at least four times faster than the Tx/Rx clock for NDUSCC and faster than the Tx/Rx clock for CDUSCC.

– TRSR[7], TxEMPTY, is not the same as TxRDY on FIFO EMPTY, GSR[5], when OMR[4]=1. In this case, GSR[5] will be asserted when the last character from the FIFO is loaded into the shift register, making the FIFO empty. TRSR[7] is asserted later, when the last available character has been completely serialized and transmitted, and both the FIFO and the shift register are empty.

– Some unused inputs on the part should not be left floating, specifically:

- When not using vectored interrupts, IACKN must be tied high
- When not using DTCN, it can be left open or tied high
- When not using DONEN, it must still have a pull–up resistor to 5V.

– A channel cannot be dynamically reconfigured. Do not write to CMR1, CMR2, S1R, S2R or PCR when the channel is in use (receiver or transmitter enabled). Do not write to RPR or RTR if the receiver is enabled. Do not write to TPR or TTR if the transmitter is enabled. And, do not write to CTCR, CTPRH or CTPRL if the counter/timer is enabled.

–The REOM status bit is about 150ns before the RxRDY status bit, so that in an interrupt driven system, if IACKN is asserted during the window, the REOM status will be recognized as the highest priority interrupt and the interrupt vector will reflect this.

– An Enable Transmitter command will be ignored if it is given after a Disable Transmitter command and there is still data in the TxFIFO. The DUSCC will wait for the TxFIFO to empty and will then disable the transmitter. A work around for this situation is to wait for the TxFIFO empty bit to be set before enabling the transmitter.

–The BISYNC protocol, when used with ASCII data, requires a Frame Check Sequence (FCS) that uses LRC7 plus odd parity. The NDUSCC/CDUSCC does not provide a LRC7 FCS. The FCS calculations need to be performed by the CPU, the actual Block Check Character will be sent and received by the DUSCC/CDUSCC as a data character in this case, the DUSCC/CDUSCC would be programmed to use no FCS (NDUSCC only).

– When using a bi–phase data encoding method (i.e., Manchester, FM0, or FM1) and an externally provided receiver baud rate clock, be aware that there is a baud rate speed limitation. For this case, the data setup time to the rising edge of the receiver clock is 120ns, while the data hold time is 10ns. Since the clock edges are usually synchronized with the center of the bit halves, this will limit you to a baud rate of just over 2Mbps (1/(120ns x 4) = 2.08Mbps). If the external clock can be skewed to make use of the short hold time required, the maximum baud late available will be just over 3.8Mbs (1/130ns x 2) = 3.85Mbps) for the DUSCC (NDUSCC only).

– In a single address DMA cycle, care should be taken not to read the Rx FIFO when the FIFO is empty. A read of the empty RxFIFO using the RTXDAKN input will cause the FIFO pointers to go out of sequence and will result in previously read data to be output onto the data bus. A ‘reset receiver’ command or a hardware reset will always set the RxFIFO pointers back to their correct initial state if the pointers have been incremented due to this erroneous access. The RxFIFO condition is indicated by the DUSCC negating the RTxDRQN output. This caution only applies to the DUSCC.

–The SCN68562 and SC68C562 do not support the use of a ‘retry’ operation during a single address DMA cycle. A 684xx DMA controller has the capability to terminate the current bus cycle and then start it again when it receives a ‘retry’ exception code. It terminates the cycle by immediately negating its DMA ACK output, it will not assert its DTC output. However, the DUSCC will assume a valid DMA cycle and will complete the DMA operation (either reading data off the bus or placing data onto the bus) when it receives DTC asserting or DMA ACK negating, WHICHEVER OCCURS FIRST.

Initialization Caution for Asynchronous Mode Local Loop Operation

This caution only applies to situations where the local loop channel connection (CMR2[7]=10) is being used with the asynchronous channel protocol (CMR1[2:0]= 111). When initializing for this mode, there must be a one bit time wait period between having three basic initialization steps done (software Tx reset, put in local loop mode, and set a Tx clock) and enabling the receiver. The software Tx reset ensures that the transmitter output is high, then the part must be in local loop mode for the connection between the transmitter and receiver to be made, and a transmitter clock must be provided to clock the transmitter output through to the receiver. This high Tx output signal will take one bit time (as determined by the Tx clock) to clock through to the receiver shift register input.

Not allowing enough time between these three steps and enabling the receiver can cause the receiver to receive incorrect data. If the state of the receiver shift register input is low when the receiver is enabled, this low signal will be interpreted as a start bit, and the receiver will start assembling a character. The receiver shift register input can be either low or high on power up due to the part’s internal logic, so this setup time is always needed.

When running at slower baud rates, it can be desirable to speed up this propagation time, since, for example, one bit time is 104µs at 9600 baud and is 20ms at 50 baud. The highest speed available with the internal baud rate generator in the part is 38.4k baud (NDUSCC only), and one bit time at this baud rate is only 26µs. So, to get the shortest propagation delay, do the following:

1. Power–up, hardware reset
2. Program CCR = H‘00’, Software Tx Reset
3. Program CMR2, to make the Local Loop connections
4. Program TTR = H‘3F’, Tx use BRG at 38.4k baud as clock
5. Additional register programming for device initialization
6. If needed, additional delay to bring total time between steps 4 and 7 to 26µs.
7. Program TTR to set up desired baud rate for the Tx
8. Program CCR = H‘00’, Software Tx reset
9. Program CCR = H‘40’, Software Rx reset
10. Program CCR = H‘02’, Enable Tx
11. Program CCR = H‘42’, Enable Rx

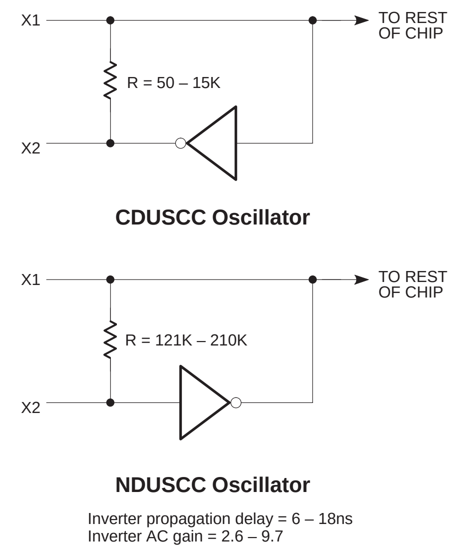
Theoretical Information on DUSCC/CDUSCC Crystal Oscillator

The information contained in Table 1 is based on computer simulations over the expected process range. It is not based on characterization data or actual device testing.

Table 1.

Parameter	Max	Typ	Min	Units
NDUSCC				
Feedback resistor[2]	210	160	121	kΩ
X1/ground capacitance	3.0	1.7	1.0	pF
X2/ground capacitance	6.0	4.3	3.0	pF
X1/X2 capacitance	2.0	1.0	0.5	pF
Inverter AC gain (14.7456MHz)[3]		2.8		dB
Inverter phase shift (14.7456MHz)[3]		249		deg.
Inverter AC gain (16MHz)[3]		2.6		dB
Inverter phase shift (16MHz)[3]		253		deg.
Inverter AC gain (2MHz)[3]		9.7		dB
Inverter phase shift (2MHz)[3]		210		deg.
X1/X2 bias level	2.9	2.3	1.8	V
Inverter prop delay[1]	18	11	6	ns
CDUSCC				
X1/ground capacitance	20	15	10	pF
X2/ground capacitance	20	15	10	pF
X1/X2 capacitance	2.0	1.0	0.5	pF
Inverter AC gain (14.7456MHz)[4]	8.5	7.3	6.0	dB
Inverter phase shift gain (14.7456MHz)[4]	260	250	240	deg.
Inverter AC gain (16MHz)[4]	7.9	7.3	6.0	dB
Inverter phase shift gain[4]	320	300	250	deg.
Inverter AC gain (2MHz)[4]	15.5	13.6	7.6	dB
Inverter phase shift (2MHz)[4]	210	190	185	deg.
X1/X2 bias level	2.9	2.3	1.8	V

NOTES:
1. 10pF load on output X1. Delay from X2 = 3V to X1 = 3V.
2. Based on I–V characteristics of depletion transistor.
3. V_{OUT}/V_{IN} at bias point. X1 10pF loading.
4. V_{OUT}/V_{IN} at bias point, X1 10pF loading.



DUSCC/CDUSCC Oscillator

User notes for the SCN68/26562 (NDUSCC) and SC68/26C562 (CDUSCC)

AN416

Definitions

Short-form specification — The data in a short-form specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.

Limiting values definition — Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.

Application information — Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors make no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Disclaimers

Life support — These products are not designed for use in life support appliances, devices or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such application.

Right to make changes — Philips Semiconductors reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

Philips Semiconductors
811 East Arques Avenue
P.O. Box 3409
Sunnyvale, California 94088–3409
Telephone 800-234-7381

print code Date of release: 02-89

Document order number: 9397 750 04199

Let's make things better.

Philips Semiconductors

PHILIPS