SCC2698B Octal universal asynchronous receiver/transmitter (Octal-UART)

AN410B

DEVICE ARCHITECTURE

The Philips Semiconductors SCC2698B Octal UART is composed of four blocks, each logically equivalent to a 2681 or 2692 DUART. Each block is composed of two channels, a counter/timer, and an interrupt control section. The channels are matched to the blocks as shown in Figure 1. The blocks are indicated by capital letters A, B, C, and D; the channels are indicated by lower-case letters a, b, c, d, e, f, g, and h. All registers act either on a block or an individual channel.

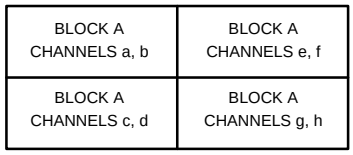

Figure 1. Channel Architecture

Registers that affect a block:
IPCR/ACR
ISR/IMR
CTU/CTUR
CTL/CTLR
IPR/OPCR
START C/T
STOP C/T

Registers that affect a channel:
MR1/MR2
SR/CSR
CR
RHR/THR

NOTE: This application note also applies to the SCC2698A Octal UART unless otherwise indicated.

X1/CLK SOURCES

The SCC2698B must have a clock source connected to the X1 input at all times. It can be supplied by a crystal between the X1 and X2 pins, or by driving an external clock into the X1/CLK input. The frequency must be between 2.0 and 4.0MHz for correct device operation; 3.6864MHz is the nominal frequency which is used to obtain the standard baud rates listed for the internal baud rate generator.

X1/X2 Crystal

The SCC2698B oscillator circuitry consists of an inverting amplifier and a feedback resistor which are used to implement a Pierce oscillator (see Figure 2). This circuitry will cause the crystal attached between the X1 and X2 pins to go into anti-resonant (parallel) operation. So, while a number of crystal and capacitor combinations will work, obtaining a parallel calibrated crystal and adjusting the external capacitor values until the total circuit capacitance matches the capacitance specified for the crystal will result in the most accurate frequency value. Using 24pF capacitors and the parallel crystal recommended below will give accurate, reliable results. The frequency will vary slightly depending on the amount of stray capacitance in the individual circuit, but will typically be off no more than 0.01%. The frequency can be adjusted by trimming the external capacitors; larger capacitors lower the oscillator's frequency and smaller ones raise it.

A source for the 3.6864MHz crystal is: Saronix, Palo Alto, CA. From California, call (800) 422-3355; outside California, call (800) 227-8974. Request part number NYP037-20.

Externally Driven Clock

The most important point in using an external source to drive the X1/CLK input is to meet the V_{IH} specification of $0.8V_{CC}$ (4.0V at V_{CC} = 5.0V). This can be insured by using an open collector buffer with a pull-up resistor to V_{CC} to drive the X1 input. Also, when driving a clock into X1, be sure to leave the X2 pin open; grounding it will kill the oscillation.

BAUD RATE GENERATION TECHNIQUES

There are 18 standard baud rates available using the internal baud rate generator when the X1/CLK frequency is 3.6864MHz. These are selected by ACR[7] and by CSR[7:4] for the receiver and CSR[3:0] for the transmitter.

The baud rate generator table follows:

Table 1. Baud Rate Generator Table

CSR[7:4] (or [3:0])	ACR[7] = 0	ACR[7] = 1
0000	50	75
0001	110	110
0010	134.5	38.4k
0011	200	150
0100	300	300
0101	600	600
0110	1,200	1,200
0111	1,050	2,000
1000	2,400	2,400
1001	4,800	4,800
1010	7,200	1,800
1011	9,600	9,600
1100	38.4k	19.2k

The baud rate generator can also be used to generate other baud rates by using a different X1/CLK frequency. For this case, each ACR[7] and CSR combination gives a different division ratio. The division ratio table follows:

Table 2. Division Ratio Table

CSR[7:4] (or [3:0])	ACR[7] = 0	ACR[7] = 1
0000	73,728	49,152
0001	33,536	33,536
0010	27,392	96
0011	18,432	24,576
0100	12,288	12,288
0101	6,144	6,144
0110	3,072	3,072
0111	3,520	1,840
1000	1,536	1,536
1001	768	768
1010	512	2,048
1011	384	384
1100	96	192

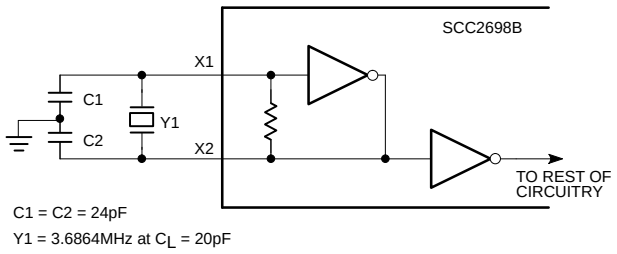

Figure 2. Pierce Oscillator Using On-Board Oscillator Circuitry

The baud rate can be calculated by dividing the X1/CLK frequency by the appropriate division ratio. For example, if the X1/CLK frequency = 3MHz, ACR[7] = 0 and CSR = CC hex, the division ratio is 96 and both the receiver and transmitter will use a 31.25K baud rate.

Externally Generated Baud Rate

An externally generated baud rate clock can be used for each transmitter and receiver using multipurpose pins MPP1 and MPP2. These are only available in the 84-pin PLCC package. OPCR[7] must be programmed to a zero to use these pins as inputs. MPP2 is used as a 16X clock source for the receiver by programming CSR[7:4] = 1110 and as a 1X clock source by programming CSR[7:4] = 1111. MPP1 is used as a 16X clock source for the transmitter by programming CSR[3:0] = 1110 and a 1X clock source by programming CSR[3:0] = 1111. (NOTE: MPP1 and MPP2 on the SCC2698B correspond to MPI2 and MPI3 on the SCC2698A. On the SCC2698A, the state of OPCR[7] does not affect the function of MPI2 and MPI3.) The maximum frequency that can be used as a 16X clock is 2MHz, which results in a baud rate of 125Kbps. The maximum frequency that can be used as a 1X clock is 1MHz, for a maximum baud rate of 1Mbps.

Counter/Timer as 16X Baud Rate Clock

The counter/timer can be used in timer mode to divide the X1/CLK or an external clock. The output of the C/T is internally connected as a 16X clock source for the receiver by programming CSR[7:4] = 1101 and as a 16X clock source for the transmitter by programming CSR[3:0] = 1101. The clock source for the timer is selected by ACR[6:4] as follows:

ACR[6:4]	Timer Clock Source
100	MPI1 pin *
101	MPI1 pin divided by 16*
110	X1/CLK
111	X1/CLK
* The MPI1 pin available as a clock source is MPI1 a, c, e, and g only	

In addition, the CTUR and CTLR registers must be programmed with the divisor value for the timer. The minimum allowable value to program is 0002 hex. The timer will generate a square wave with a period of twice the number of timer clock periods programmed into CTUR/CTLR. The resultant baud rate is calculated by:

$$\left(\frac{\text{timer clock}}{2 \times \text{CTUR/CTLR value}}\right) \div 16$$

The maximum baud rate available in this manner is 62.5Kbps. This is obtained with an X1/CLK = 4MHz and programming as shown in example 1.

Counter/Timer as 1X Baud Rate Clock

The timer can also be used as a 1X clock source for the transmitter by externally connecting the C/T output to MPP1 (MPI2 on the SCC2698A). In addition, the C/T output can be connected to MPP2 (MPI3 on the SCC2698A) to provide a 1X clock source for the receiver, but care must be taken to have the timer output synchronized with the incoming data to ensure accurate data reception. These inputs are only available in the 84-pin PLCC packaged part. The C/T is set up as described in the last section. The resultant baud rate is calculated by:

$$\left(\frac{\text{timer clock}}{2 \times \text{CTUR/CTLR value}}\right)$$

The maximum baud rate available in this manner is 1Mbps. This is obtained with an X1/CLK = 4MHz, MPO externally connected to MPP1 and MPP2 and programmed as shown in example 2.

RTS/CTS FLOW CONTROL

One way to achieve flow control with the SCC2698B is to have the request to send (RTS) output, controlled by the receiver, connected to the clear to send (CTS) input, which enables the transmitter. RTS is controlled by the receiver when MR1[7] = 1, and the multi-purpose output (MPO) is used as the RTS output when OPCR[6:4] = 0 and OPCR[2:0] = 0.

Initially, the RTS output must be asserted by writing CR[7:4] = 1000 immediately after enabling the receiver. After this, RTS will automatically negate upon receipt of a valid start bit if the receiver FIFO is full, and will reassert when an empty FIFO position is available. CTS enables the transmitter and MPI0 is used as the CTS input pin when MR2[4] = 1. When CTS is negated, the transmitter will complete transmitting a character already in progress, but will not transmit a character waiting in the THR. The TxD output will then go into the marking state and the transmitter clock will be stopped. The Tx empty bit will not be set (even if the transmitter is empty) until the transmitter clock starts running again. When CTS is re-asserted, the transmitter will start again, transmitting if a character is waiting in the THR or setting the empty bit if not. Be careful if the transmitting device is not a Philips Semiconductors part, as some devices will transmit both the character in the transmitter shift register and the character in the transmitter holding register when CTS is negated. If this occurs, the receiver may be overrun by a fifth character.

Modem control configurations may require RTS to be controlled by the transmitter, asserted for the entire time the message is being sent, and negated on completion of the transmission. RTS is controlled by the transmitter when MR2[5] = 1, and MPO is used as the RTS output when OPCR[6:4] = 0 and OPCR[7:4] = 1000 after the transmitter has been enabled and before the first byte of the message is loaded into the THR. This last character will be transmitted and RTS will be negated one bit time after the last stop bit.

Example 1.

```
CSD=DD HEX          ;Rx & Tx USE TIMER AS
                     16X BAUD RATE
                    ;CLOCK
ACR[7]=DON'T CARE
ACR[6:4]=110        ;X1/CLK IS TIMER 1X CLOCK
                     SOURCE
CTUR/CTLR=0002 HEX;TIMER HAS DIVISOR OF 2
```

Example 2.

```
CSR=FF HEX          ;Rx USES MPP2 AND Tx USES
                    ;MPP1 AS A 1X BAUD RATE CLOCK
ACR[7]=DON'T CARE
ACR[6:4]=110        ;X1/CLK IS TIMER 1X CLOCK
                    ;SOURCE
CTUR/CTLR=0002 HEX;TIMER HAS DIVISOR OF 2
OPCR[7:4]=0001      ;MP0 IS C/T OUTPUT, MPP1 AND
                    ;MPP2 ARE CLOCK INPUTS
```

Note that this example is shown to further demonstrate the device's versatility in baud rate generation. For most applications, this method is good for the transmit clock, but may not be a recommended method for accurate data reception, unless the timer clock source is synchronized with the incoming data.

Example 3.

```
RXRDY:  MOVE.B   #$00,$3000  ; DUMMY WRITE
        MOVE.B   SR,D3       ; READ STATUS REGISTER
        BTST     #0,D3       ; CHECK FOR RXRDY
        BEQ      RXRDY       ; IF NOT, KEEP CHECKING
```

MULTI-PURPOSE INPUTS

There are four multi-purpose inputs provided for each channel for the 84-pin PLCC package, MPI0 and MPI1, and MPI1 and MPI2 when OPCR[7] is programmed as a zero. The DIP package has one multi-purpose input for each channel, MPI0. The current state for each of these pins can be read from the input port register (IPR). Each input can be used as a general purpose input, to be interpreted as the user desires. In addition, each input can be programmed to provide a specific defined input, as follows:

– Current state also in the input port change register (IPCR)
– Has a change of state indicator in IPCR, which can also be used to generate an interrupt.
– When MR2[4] = 1, it is used as CTSN input to enable the transmitter.

MPI1

– Current state also in the input port change register (IPCR).
– Has a change of state indicator in IPCR, which can also be used to generate an interrupt.
– When ACR[6:4] = 000, MPI1a, MPI1c, MPI1e and MPI1g are used as a 1X counter clock source. When ACR[6:4] = 001, MPI1a, MPI1c, MPI1e and MPI1g are used as a 16X timer clock source.
– For all four of these programmed cases, MPI1b, MPI1d, MPI1f and MPI1h stay as general purpose inputs, since there is only one C/T clock for each block.

MPP1 (MPI2 on SCC2698A)

– When CSR[3:0] = 1110, it is used as the transmitter 16X baud rate clock input.

MPP1 (MPI2 on SCC2698A)

– When CSR[7:4] = 1110, it is used as the receiver 16X baud rate clock input.
– When CSR[7:4] = 1111, it is used as the receiver 1X baud rate clock input.

Rx AND Tx STATUS OUTPUTS

When OPCR[7] of a block is programmed to '1', the MPP pins for that block become active low open drain outputs. MPP1 then becomes the TxRDY status output for its corresponding transmitter, while MPP2 then becomes the RxRDY/FFULL status output for its corresponding receiver. These outputs may be used for interrupts or as DMA request control signals. These outputs are not masked by the IMR register. NOTE: This function is not available on the SCC2698A.

BUS INTERFACE

The internal control signals for strobing read and write cycles are obtained by internally logically ANDing CEN with RDN, and CEN with WRN. As a consequence, the signal asserted last initiates the cycle and the signal negated first terminates the cycle. However, the RDN line cannot be left asserted with just the CEN line pulsed. The RDN signal must be negated between reads, because the status register can only be updated at that time.

When using a 68000 type bus interface with a static R/WN output, either a hardware or a software method of pulsing the RDN line must be designed in. An example of a hardware solution is to logically AND the CEN signal with the inversion of R/WN to provide RDN (see attached schematic). RDN can also be negated by the software, the inversion of R/WN can be used for RDN, and a dummy write can be performed between consecutive reads. For example, in a polling loop of the status register, see example 3.

POWER DOWN MODE

The 2698B is equipped with a special power down feature which can be used for energy conservation during idle periods. This mode saves the contents of all the internal registers, stops the oscillator and suspends the operation of any function that uses the oscillator. In addition, the I_{CC} current used by the part is reduced to 2mA. The part can be put into power down mode at any time, and restored to normal operation when needed. Since all register values are saved, reinitialization is not necessary.

To put the part into power down mode:

For each channel:
– CR7:4] = 0011 – Reset Tx
– CR[7:4] = 0010 – Reset Rx

Once only:
– OPCRA[3] = 0 – Turn off power down mode

To get out of power down mode:

Once only:
– OPCRA[3] = 0 – Turn off power down mode

For each channel:
– CR[3:0] = 0101 – Enable Rx and Tx

GENERAL INITIALIZATION

Figure 3 shows the flow of a typical initialization. Note that the transmitter and receiver should be disabled before writing to MR1,

MR2, CSR, ACR[7] or OPCR[3] and reset before continuing operation. The other registers, if used, can be changed at any time. A local loopback mode program example is also given to demonstrate the basic initialization and use of the device. This example checks each channel separately, each is initialized in local loopback mode and uses polled operation to send and receive 256 characters, comparing each character received to the one that was sent.

WAKE-UP MODE

The wake-up mode provides automatic wake-up of the receiver through address frame recognition for multi-processor communications. This mode is selected by programming MR1[4:3] = 11. In this mode, the transmitter (in the primary or master station) will send data with the last bit of each character identified as the address/data (A/D) bit. When MR1[2] = 0, the A/D bit will be transmitted as a '0', which will cause the character to be interpreted by the receivers (in the secondary or slave stations) as a data character. When MR1[2] = 1, the A/D bit will be transmitted as a '1', which will cause the character to be interpreted by the slave stations as an address character. The slave stations are normally disabled, but the receivers monitor the received data stream and will load the character into the RHR FIFO and set the RxRDY status bit if the character has the address bit set. The slave station can then examine the address character to see if it should read the subsequent data.

The master station should allow adequate time between sending the address character and the first data character for the slowest slave station to determine if it has an address match. Once a station has an address match, it then enables its receiver to start receiving data.

Some applications have to interface to an existing system, and have no control of the time between the address and data characters. When this time is too short for the CPU to make an address verification, the first data character could be missed. This would generally only be a problem at higher baud rates (192.k, 38.4k). If this is a problem in your application, the CPU can enable the receiver as soon as the address is received. Later, after the compare operation, the CPU can either read the data or reset the receiver depending on whether the address was a match.

Changing between address and data modes in the master station requires a write to MR1. Before doing this, wait for transmitter empty to indicate that the previous character is done and then disable the transmitter and receiver of this station. See Figures 4 and 5, and the wake-up mode program example.

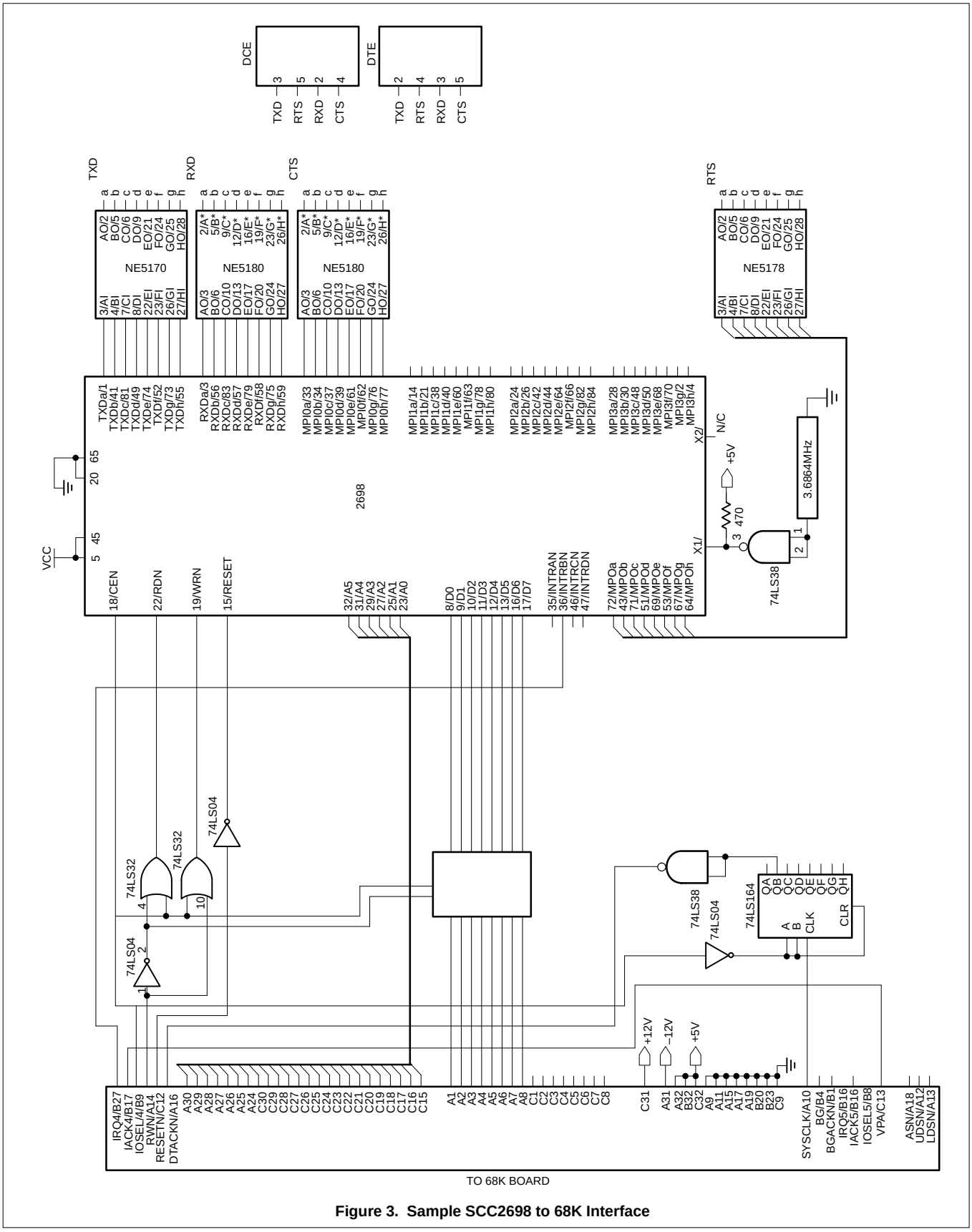

Figure 3. Sample SCC2698 to 68K Interface

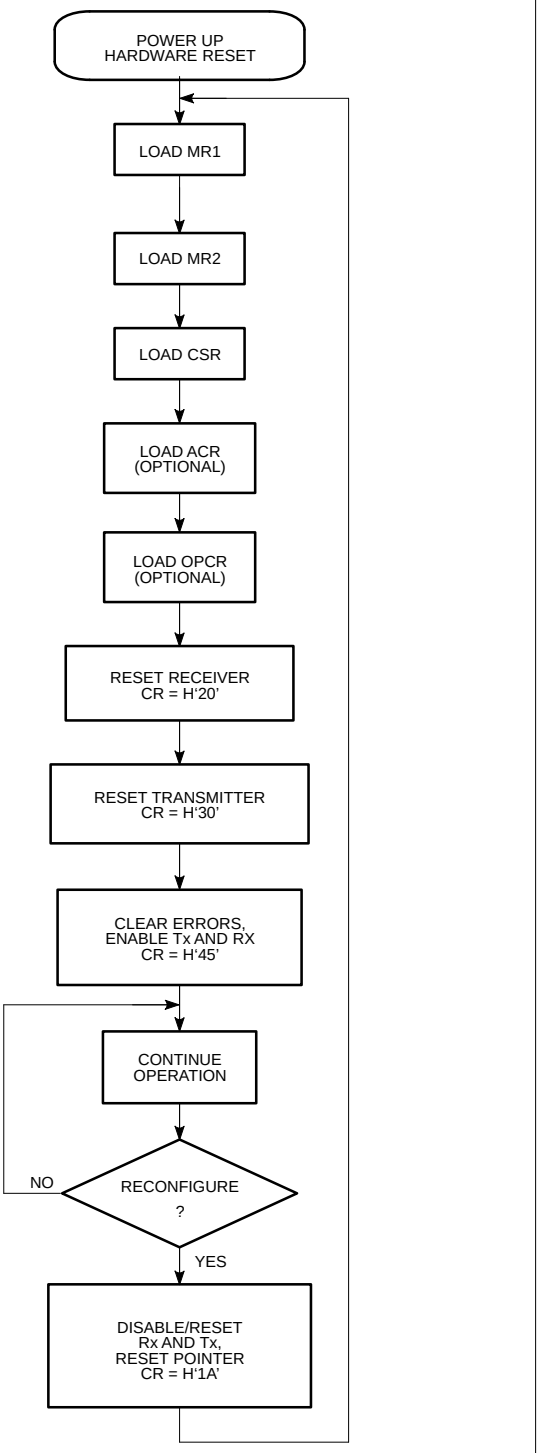

Figure 4. Basic Initialization

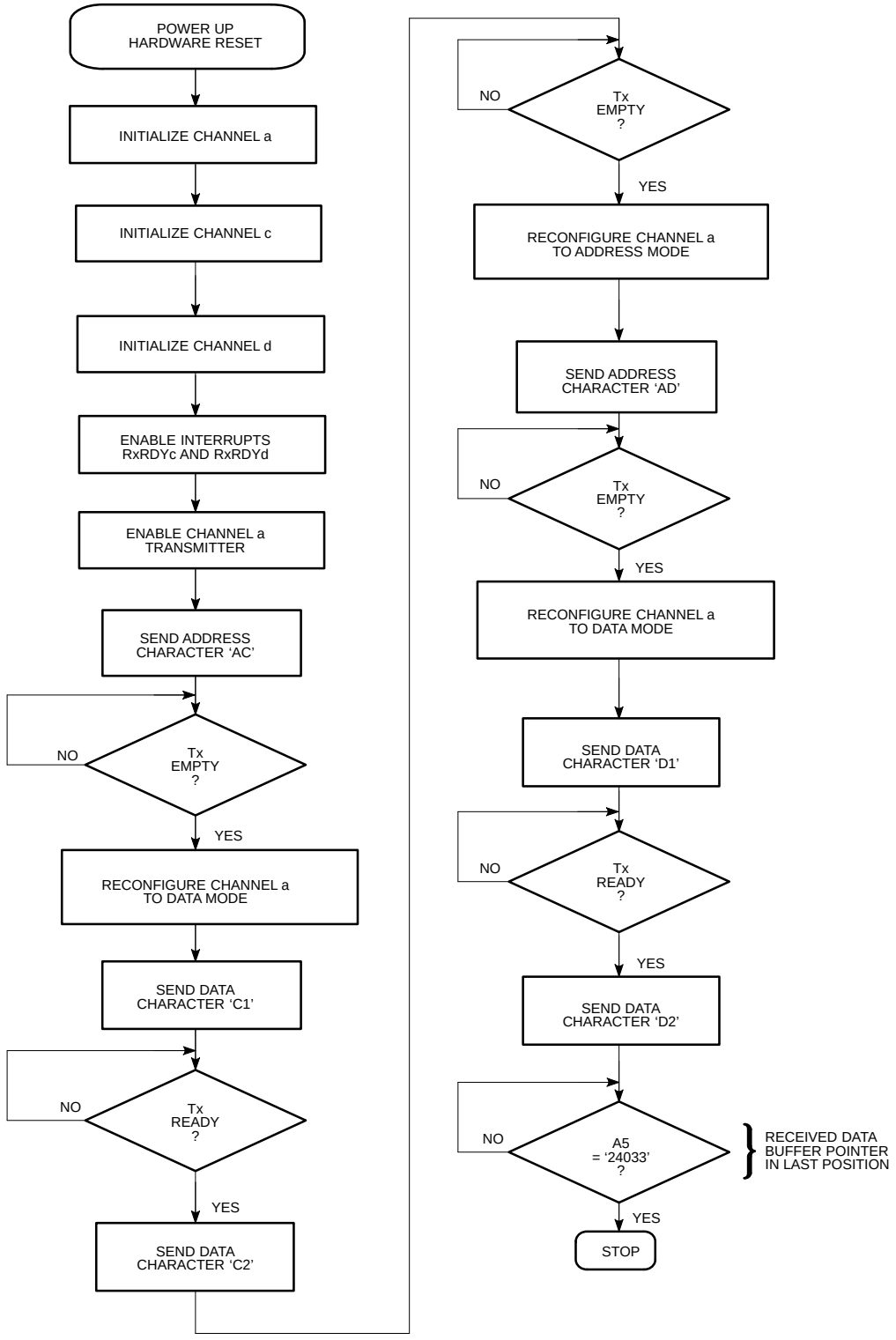

Figure 5. Wake-Up Mode Example, Main Program Flowchart

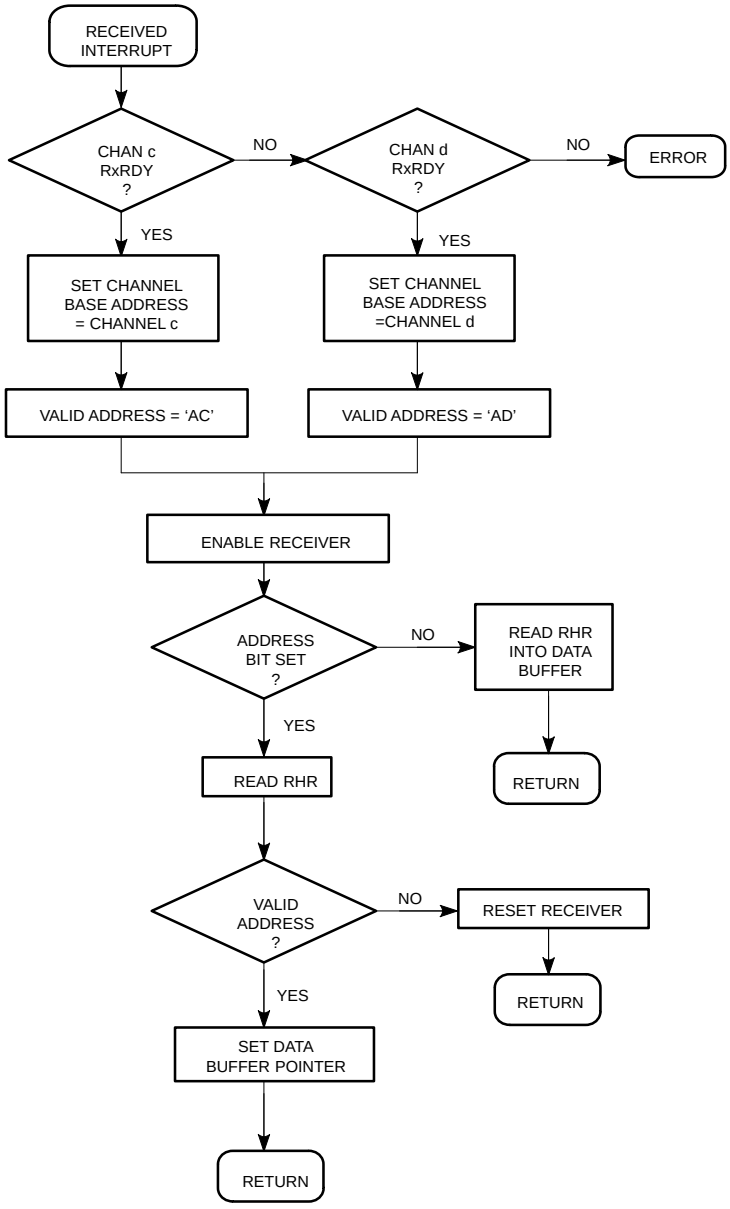

Figure 6. Wake-Up Mode Example, Interrupt Routine Flowchart

LOCAL LOOPBACK MODE PROGRAM EXAMPLE

```
;THIS IS PROGRAM 2698 LOCAL LOOP
;IT SEPARATELY TESTS EACH CHANNEL IN LOCAL LOOPBACK
;MODE, SENDING 256 CHARACTERS AND COMPARING EACH
;CHARACTER SENT WITH EACH ONE RECEIVED.
;D. IBARRA JULY 1987
;               BEGIN
CHANa           EQU         $74001          ;CHANNEL BASE ADDRESSES
CHANb           EQU         $74011
CHANc           EQU         $74021
CHANd           EQU         $74031
CHANe           EQU         $74041
CHANf           EQU         $74051
CHANg           EQU         $74061
CHANh           EQU         $74071
;
BLOCKA          EQU         $74001          ;BLOCK BASE ADDRESSES
BLOCKB          EQU         $74021
BLOCKC          EQU         $74041
BLOCKD          EQU         $74061
;
MR1             EQU         $0              ;CHANNEL REGISTER OFFSETS
MR2             EQU         $0
STATR           EQU         $2
CSR             EQU         $2
CR              EQU         $4
RHR             EQU         $6
THR             EQU         $6
;
IPCR            EQU         $8              ;BLOCK REGISTER OFFSETS
ACR             EQU         $8
ISR             EQU         $A
IMR             EQU         $A
CTU             EQU         $D
CTUR            EQU         $D
CTL             EQU         $E
CTLR            EQU         $E
IPR             EQU         $1A
OPCR            EQU         $1A
STRTCT          EQU         $1C
STOPCT          EQU         $1E
;
START:          MOVEA.L     #CHANa,A2       ;TEST CHAN a
                JSR         INIT
                JSR         TEST
                MOVEA.L     #CHANb,A2       ;TEST CHAN b
                JSR         INIT
                JSR         TEST
                MOVEA.L     #CHANc,A2       ;TEST CHAN c
                JSR         INIT
                JSR         TEST
                MOVEA.L     #CHANd,A2       ;TEST CHAN d
                JSR         INIT
                JSR         TEST
                MOVEA.L     #CHANe,A2       ;TEST CHAN e
                JSR         INIT
                JSR         TEST
                MOVEA.L     #CHANf,A2       ;TEST CHAN f
                JSR         INIT
                JSR         TEST
```

```
                MOVEA.L     #CHANg,A2       ;TEST CHAN g
                JSR         INIT
                JSR         TEST
                MOVEA.L     #CHANh,A2       ;TEST CHAN h
                JSR         INIT
                JSR         TEST
;
                MOVEA.L     #0,A2           ;CLEAR ADDRESS REG.
STOP:           TRAP        #15
;
;———————————SUBROUTINES———————————————————
;
INIT:           MOVE.B      #$1A,CR[A2]     ;DISABLE TX & RX, RESET POINTER
                MOVE.B      #$13,MR1[A2]    ;NO PARITY, 8 BITS
                MOVE.B      #$87,MR2[A2]    ;LOCAL LOOP, STOP=1
                MOVE.B      #$66,CSR[A2]    ;TXC=RXC=BRG,1200 BAUD
                MOVE.B      #$20,CR[A2]     ;RESET RECEIVER
                MOVE.B      #$30,CR[A2]     ;RESET TRANSMITTER
                MOVE.B      #$45,CR[A2]     ;CLR ERRORS, ENABLE TX–RX
                RTS
;
TEST:           MOVE.W      #$100,D7        ;CLEAR SEND REGISTER
AGAIN:          SUBI.B      #$1,D7          ;DEC. SEND CHAR.
                BEQ         DONE            ;UNTIL D7=0, THEN DONE
                MOVE.B      D7,THR[A2]      ;TRANSMIT CHAR
                BSR         RXRDY           ;WAIT FOR RXRDY
                MOVE.B      RHR[A2],D1      ;RECEIVE CHAR INTO D1
                CMP.B       D7,D1           ;SENT CHAR=RECEIVE CHAR ?
                BEQ         AGAIN           ;IF SO, KEEP SENDING
                TRAP        #15             ;STOP IF FAIL
DONE:           RTS
;
RXRDY:          MOVE.B      STATR[A2],D3    ;STATUS REG. TO D3
                BTST        #0,D3           ;IS RECEIVER READY ?
                BEQ         RXRDY           ;IF NOT, WAIT
                RTS

                END         START
```

wake-up MODE PROGRAM EXAMPLE

```
;THIS IS PROGRAM 2698 WAKE-UP
;IT USES CHAN. a AS THE MASTER STATION, AND
;CHANNELS c AND d AS SLAVE RECEIVING STATIONS.
;CHAN. a TRANSMITTER IS EXTERNALLY CONNECTED
;TO c AND d RECEIVERS.  THIS PROGRAM IS INTERRUPT
;DRIVEN, THE SLAVE STATIONS WILL INTERRUPT ON RXRDY.
;
;D. IBARRA JULY 1987
;               BEGIN
;
CHANa       EQU         $74001          ;CHANNEL BASE ADDRESSES
CHANb       EQU         $74011
CHANc       EQU         $74021
CHANd       EQU         $74031
CHANe       EQU         $74041
CHANf       EQU         $74051
CHANg       EQU         $74061
CHANh       EQU         $74071
;
BLOCKA      EQU         $74001          ;BLOCK BASE ADDRESSES
BLOCKB      EQU         $74021
BLOCKC      EQU         $74041
BLOCKD      EQU         $74061
;
MR1         EQU         $0              ;CHANNEL REGISTER OFFSETS
MR2         EQU         $0
STATR       EQU         $2
CSR         EQU         $2
CR          EQU         $4
RHR         EQU         $6
THR         EQU         $6
;
IPCR        EQU         $8              ;BLOCK REGISTER OFFSETS
ACR         EQU         $8
ISR         EQU         $A
IMR         EQU         $A
CTU         EQU         $D
CTUR        EQU         $D
CTL         EQU         $E
CTLR        EQU         $E
IPR         EQU         $1A
OPCR        EQU         $1A
STRTCT      EQU         $1C
STOPCT      EQU         $1E
;
START:      MOVEA.L     #CHANa,A1
            MOVEA.L     #CHANa,A2       ;INIT CHAN a
            JSR         INIT
            MOVEA.L     #CHANc,A2       ;INIT CHAN c
            JSR         INIT
            MOVEA.L     #CHANd,A2       ;TEST CHAN d
            JSR         INIT
            MOVEA.L     #BLOCKB,A3      ;ENABLE RXRDY INTERRUPTS
            JSR         SETINT
;
            MOVE.B      #$04,CR[A1]     ;ENABLE TX CH. a
            MOVE.B      #$AC,THR[A1]    ;SEND ADDRESS CHAR
            JSR         DTMODE          ;CHANGE TO DATA MODE
```

```
                MOVE.B      #$C1,THR[A1]    ;SEND DATA
                JSR         TXRDY           ;WAIT FOR TXRDY
                MOVE.B      #$C2,THR[A1]    ;SEND DATA
                JSR         ADMODE          ;CHANGE TO ADDRESS MODE
                MOVE.B      #$AD,THR[A1]    ;SEND ADDRESS CHAR
                JSR         DTMODE          ;CHANGE TO DATA MODE
                MOVE.B      #$D1,THR[A1]    ;SEND DATA
                JSR         TXRDY           ;WIAT FOR TXRDY
;
WTDN:           MOVE.L      A5,D1           ;MOVE DATA BUFFER POINTER TO D1
                CMP.L       #$24033,D1      ;IS IT IN LAST POSITION ?
                BNE         WTDN            ;IF NOT, WAIT UNTIL DONE
                MOVEA.L     #0,A2           ;CLEAR ADD. REG.
                TRAP        #15             ;TO INDICATE NORMAL END
;
;————————————SUBROUTINES————————————————————
;
INIT:           MOVE.B      #$1A,CR[A2]     ;DISABLE TX & RX, RESET POINTER
                MOVE.B      #$1F,MR1[A2]    ;WAKE-UP, 8 BITS, A/D=1
                MOVE.B      #$07,MR2[A2]    ;NORMAL, STOP=1
                MOVE.B      #$66,CSR[A2]    ;TXC=RXC=BRG,1200 BAUD
                MOVE.B      #$20,CR[A2]     ;RESET RX
                MOVE.B      #$30,CR[A2]     ;RESET TX
                MOVE.B      #$40,CR[A2]     ;CLEAR ERRORS
                RTS
;
SETINT:         LEA.L       INT,A6          ;PUT INT. ROUTINE ADD. INTO A6
AGAIN:          MOVE.L      A6,$10476       ;ADD. TO AUTO VECTOR #4 LOCATION
                MOVE.B      #$22,IMR[A3]    ;INT. ON RXRDY BOTH CHANNELS
                RTS
;
DTMODE:         JSR         TXEMTY          ;WAIT FOR TXEMPTY
                MOVE.B      #$1A,CR[A1]     ;DISABLE TX AND RX, RESET POINTER
                MOVE.B      #$1B,MR1[A1]    ;WAKE-UP, 8 BITS, A/D=0
                MOVE.B      #$20,CR[A1]     ;RESET RX
                MOVE.B      #$30,CR[A1]     ;RESET TX
                MOVE.B      #$44,CR[A1]     ;CLEAR ERRORS,ENABLE TX
                RTS
;
ADMODE:         JSR         TXEMTY          ;WAIT FOR TXEMPTY
                MOVE.B      #$1A,CR[A1]     ;DISABLE TX AND RX, RESET POINTER
                MOVE.B      #$1F,MR1[A1]    ;WAKE-UP, 8 BITS, A/D=1
                MOVE.B      #$20,CR[A1]     ;RESET RX
                MOVE.B      #$30,CR[A1]     ;RESET TX
                MOVE.B      #$44,CR[A1]     ;CLEAR ERRORS,ENABLE TX
                RTS
;
TXEMTY:         MOVE.B      STATR[A1],D1    ;MOVE STATUS REG. TO D1
                BTST        #3,D1           ;IS TX EMPTY ?
                BEQ         TXEMTY          ;IF NOT, WAIT
                RTS
;
TXRDY:          MOVE.B      STATR[A1],D1    ;MOVE STATUS REG. TO D1
                BTST        #2,D1           ;IS TXRDY ?
                BEQ         TXRDY           ;IF NOT, WAIT
                RTS
;
```