APPLICATION NOTE

AN405

SCN2681/SCN68681 and SCC2691 data communications

Supersedes data of 1986 Aug

1998 Sep 21

Philips
Semiconductors

PHILIPS

INTRODUCTION

This SCN2681/SCN68681 and SCC2691 data communications applications note contains answers to some of the most frequently discussed user inquiries. There are three main sections: functions that are common to all three; functions that are unique to the SCN2681/SCN68681; and a typical SCC2691 application using the musical instrument digital interface (MIDI).

DUART/UART COMMON FUNCTIONS

Reading Reserved Register

Performing a bus read operation at location 02H or 0AH will force these devices into a diagnostic operation. This diagnostic mode is used to test the baud rate generator circuitry. When a read operation occurs on either address, the device will output clock pulses on the general purpose outputs that are a multiple of the frequencies in the baud rate table.

This mode may be entered accidentally, in some cases, by a monitor program that uses a write followed by an automatic read/verify cycle. For example, a write to CRA or CRB with this type of monitor will invoke the reserved test mode. Care must be taken when a development system is used in the manual mode, since most development systems use a write followed by a read/verify cycle. Mostly, this anomaly can occur in the SC68681. If the rising edge of the write pulse occurs before the rising edge of CEN, the reserved mode is invoked (even if R/WN rising is only 10 to 20ns early). Users of the SCN68681 must be sure that the rising edge of R/WN rises with or after the rising edge of CEN.

Receiver FIFO

All three devices have a three-deep receive FIFO. The FIFO acts more like a circular queue than a FIFO. The FIFO acts more like a circular queue than a FIFO. The receiver has both a head and tail pointer. The head pointer is controlled by a bus read operation and is bumped to the next location whenever a read of the receiver takes place. The tail pointer is bumped whenever a character is assembled in the receive shift register and transferred to the receive holding register.

After an external reset is applied or a reset receiver command is issued, the head and tail pointers are at the same location in the FIFO. Although the data sheet specifies the receiver is flushed when a reset receiver command is issued, nothing is done to the contents of the receiver. Therefore, three consecutive reads of the the receiver will move the head pointer around in a circle until it comes back to the starting point. If no new data has been received in the receive shift register, the old data will still be in the FIFO.

Care must be taken when using a monitor in the manual mode, since the receiver head pointer can be bumped by a write to the transmit holding register (THR). (Write followed by a read/verify operation at the same address has already been discussed. See Reading Reserved Registers.)

The best way to determine if the receiver should be read is to poll the RxRDY bit in either the ISR or the SR registers. If RxRDY = 1, read the receiver again. Continue this loop until RxRDY = 0. Once this state is reached, stop reading the receiver, or the pointers will be bumped beyond the current valid data.

Detecting the End of Break

Detecting a break is a simple function built into all three devices. The receiver continuously samples RxD. If a low is sensed for the start bit and the full number of programmed bit times, a zero character is accumulated in the receive FIFO. If no stop bit is sensed (a mark condition, then the receiver samples beyond the character frame for one more bit time. If a low is sensed, a framing error has been detected. Once the framing error bit is set in the status register and a zero is accumulated in the receive FIFO, the resulting condition forces the received break bit to set in the ISR and in the SRA or SRB. In this manner, a start of break is detected.

In order to detect an end of break, the delta break bits in the ISR register must be tested. Whether the CPU is polling the ISR or if the CPU is interrupt driven, the zero character in the receive FIFO should not be read nor should the receive break be cleared in SRA or SRB until the break is completely over. To detect an end of break, the CPU should issue a reset delta break command (50H to CRA or CRB), which will reset the delta break bit in the ISR. If the CPU is using interrupts, the next step is to mask on the delta break interrupt. If the CPU is polling, it should continue polling until the delta break sets. Delta break will be set and interrupt only when a change of state occurs on RxD. When the rising edge of RxD occurs and the break is over, the delta break bit can be cleared (50H to CRA or CRB), the received break and framing error can be cleared (40H to CRA or CRB), and the zero character can be read from the receiver and discarded.

Disabling the Transmitter After a Short Frame

The data sheet states that the transmitter may be disabled after the last character is loaded in the transmit shift register. This is used to end a block transmission or to negate RTS, after the last character is shifted out of the transmit shift register. This method of disabling the transmitter is essential if the RTS handshake lines are to be used. It is not a good method to use when short, one or two byte frames are to be transmitted as a response to a primary or secondary station. The problem occurs when the transmitter is re-enabled to send another short frame response. If the last character of a message is still being serialized when an enable transmitter command is executed, the serialized character will either be garbled or lost.

For example, assume that the last character of a two byte frame was loaded into the THR. If the transmitter is running at 9600 baud, it will take from 1 to 2ms before the last character is completely shifted out on TxD (the 2ms time occurs when the first character in the two byte frame is currently in the TSR). If the CPU needs to send another response, it will start by enabling the transmitter and loading the first byte of the next frame. Even if the second byte of the last frame is now in the TSR, the transmitter must go to mark due to the re-enable command and the last character is either lost or garbled.

The best way to avoid this problem is to wait until the last character is completely shifted out of the TSR before disabling the transmitter. This can be verified by polling the TxEMPT bit until it is set in SRA or SRB. As pointed out earlier, this will not work when using RTS, since the RTS output will only toggle if the transmitter is disabled while the last data character is in the THR. In that case, the user can time out a delay before re-enabling the transmitter.

Disabling the Transmitter and/or Receiver on the Fly

Problems have been encountered when trying to write to the mode registers (MR2, MR1), or the clock select registers (CSRA/B), or to ACR[7] without first disabling the transmitter and receiver. If the mode register changes while character serialization is still active, the transmission may start over under the new mode configurations. If the mode register changes after serialization is complete and the transmitter is empty, two potential problems can appear: TxD may

go to the space condition for a short period of time, or TxRDY in the ISR will set and then reset. These results are obtained independent of the data written to the mode registers. Programming the registers to the current values can still product the above results.

The problem is more serious when writing to CSRA/B or ACR[7], when the clock is still running. If the transmit and/or receive clocks are changed without disabling the transmitter and receiver, a clipped or shortened clock may appear during the change from one frequency to another. These short clock pulses can lock up the transmitter and/or receiver, until they are re-enabled by command to CRA or CRB.

The best way to avoid these problems is to disable the transmitter and receiver, before changing either the mode or CSR registers or ACR[7]. Disabling the transmitter and receiver will stop the TXC and RXC while the changes are made. When the changes are complete, the transmitter and receiver should be reset and then re-enabled by software command. A reset command insures that everything is in a known state before the changes are initiated.

Setting Up the Counter/Timer as a Timer

The maximum frequency that can be generated by the C/T (counter/timer) is dependent on whether the 1X or 16X clock source mode is used. If the C/T is to be used to drive the receiver directly, a 16X clock must be generated that will sample the incoming data stream.

As an example, let the X1/CLK input = 4MHz and set the C/T to its minimum value (CTUR/CTLR = 0002H). Since the C/T will count down to zero before its output changes state, two full counts must pass before the C/T output changes back to its original state. Therefore, the maximum frequency output will be 4MHz divided by 4 (minimum count) divided by 16 (for sampling clock). This results in a 62.5kHz baud rate.

If a higher C/T clock rate is required, the 1X mode must be programmed. The highest C/T output is 1MHz (4MHz divided by 4). The C/T can be programmed to output 1MHz on OP3, which can be tied by wire to IP3/4 or IP5/6. These inputs can be selected to be transmit and receive 1X clock inputs. To program this type of function, write OPCR = 04H, ACR = 60H, CTUR/CTLR = 0002H, and CSRA/B =FFH. Start the C/T by performing a read at address 0EH (start counter command). When in the timer mode, the C/T will not stop oscillating until ACR[6:4] is written to the counter mode followed by a stop time command (read at address 0FH).

The counter can be used to count external events, or used as a system delay timer. As an example, the C/T can be set up to time out a 2ms delay and interrupt the CPU (used to refresh dynamic RAM). Using the X1/CLK at 4MHz as the C/T clock, the total time in counts is (2ms) / (250ns times 16) = 500 or 01F4H (times 16 is used since X1/CLK divided by 16 is the only internal clock source available in the counter mode). This function would be implemented by writing ACR = 30H, CTUR/CTLR = 01F4H, and IMR = 04H. The counter must be stopped and started by command from the CPU for each count down (stop = read at address 0FH, start = read at address 0EH).

Multidrop/Wake-Up Mode

If MR1[4:3], the devices are in the multidrop or wake-up (SCC2691) mode. This will cause the transmitter to send data with the last bit of each character identified as the address/data (A/D) bit. If MR1[2] = 0, the A/D bit will be ‘1’ and the assembled character will be interpreted by the secondary receiver as an address. Both primary and secondary stations must be in the multidrop mode. In order to transmit an address character followed by data characters, a write must be performed to MR1. Writing of the mode registers may cause garbled data, if the transmitter and receiver are not disabled. The following sequence should be used when writing to the mode registers:

1. To insure that the transmitter is in a quiescent state, do not write to MR1 until TxEMT = 1.
2. Reset the MR pointers. Disable the transmitter and the receiver.
3. Write MR1 using the previously written data, with MR1[2] = 1 (Tx address). Reset and enable the transmitter and receiver.
4. Load the address character in the transmit holding register (THR). The A/D bit will be appended to the character during transmission according to the polarity of MR1[2].
5. Wait until TxEMT = 1. (Wait for transmitter empty).
6. Reset the MR pointers. Disable the transmitter, and the receiver.
7. Write MR1 using previous data with MR1[2] = 0 (Tx data). Reset and enable the transmitter and receiver.
8. Load the first data character into the THR. Continue sending data until the message is done. To send a message to a different address, repeat steps 1 through 8.

When the secondary station sets RxRDY = 1, the CPU must immediately read the receiver to determine if the address is correct. If the received address compares, the CPU must set RxEN = 1 in CRA or CRB so the message can be received. At higher baud rates (19.2k and 38.4k), some users have lost the first character of the message. This is due to the amount of time required for the CPU to read the address out of the receiver and finish a compare operation before enabling the receiver. If this is a problem, the CPU can enable the receiver (RxEN = 1) as soon as the address is received and the CPU is initially interrupted. Later, after the compare operation is finished, the CPU can either read the data from the receiver or reset the receiver depending on the received address (see Figure 1 for a software example).

Handling Interrupts

When the transmitter is enabled and TxRDY is masked on in the IMR, and interrupt will occur immediately. If no messages are to be sent, the user should mask TxRDY off (IMR[4:0] = 0), or no other interrupts will be generated. Only the interrupting section of the device can reset the current interrupt.

For example, loading the transmitter resets TxRDY, reading the receiver resets RxRDY, stopping the counter resets counter ready, etc. If the function is not fully serviced, all other pending interrupts are held up. To avoid this problem only enable the IMR bits that will be immediately in use. Enable and disable TxRDY in the IMR just prior to and at the end of a block transmission.

Driving X1 Externally or Using a Crystal

If a user wants to use an external clock instead of a crystal, the best way is to drive X1 and ground X2 (SCN2681/68681 only). The data sheets show two inverters used to drive X1 out of phase with X2. While this is acceptable, it sacrifices the use of one gate. The only drawback to driving the clock inputs from an external source is that a minimum high voltage of 4.0 volts is required. A V_{OH} greater than 4.0 volts can be insured by use of an open collector buffer (with resistor), or by adding a pull-up resistor to an ordinary TTL buffer (be sure V_{OL} is less than 0.8 volts). Another requirement is the minimum high and low clock pulse width must be 100ns. This parameter can best be met with an external oscillator that has 50% duty cycle.

Another more subtle problem has been seen using a crystal on the X1/X2 inputs. If capacitors C1 and C2 with values of 15pF or greater are used, power-on problems may be experienced

intermittently. The crystal may not oscillate due to an insufficient charge on the capacitors. It is recommended that C1 and C2 be around 5pF to insure proper charging during the power on cycle. Many DUARTs show a 60/40 duty cycle when the crystal is installed between X1 and X2. When viewed with an oscilloscope on X1, the typical high time is 90ns, while the low time is approximately 130ns. To force the crystal to operate at a 50% duty cycle (with clock high and low time approximately 110ns), a 100kΩ or greater resistor should be added from the X1 input to the X2 input. This addition will raise the oscillator's trip point and force the crystal high and low times to be equal.

In the case of the SCC2691, none of the above problems have been experienced with the crystal. The SCC2691 is different when driven from an external source. Any time X1 is driven, X2 must be left open. If X2 is grounded, the clock will not oscillate. Note that the X1 output can only drive one CMOS external buffer. Care must be taken not to overload the X1/CLK input.

X1/X2 Crystal

When ordering the 3.6864MHz crystal, either a series or a parallel crystal can be used as long as the frequency tolerance is close to +.005%. Because of the nature of the X1/X2 circuit, a parallel crystal should be used. Testing has shown that if the tolerance value is low, the error in frequency is divided down to the point where it is negligible. A series crystal can be used if the tolerance is +.005% or less. For crystal samples call: Saronix, located in Palo Alto, CA. Request part no. NMP037: 3.6864MHz HC-18/U. A second crystal source is U.S. Crystal, located in Fort Worth, Texas. Request part number SIG36864-HC18.

General Initialization

Figure 2 describes the typical flow of software initialization. Usually the mode registers are first. If a reset was issued prior to the initialization, there is no need to reset the MR pointers. Note that the transmitter and receiver should be disabled and reset when either the mode registers (MR1A/B and MR2A/B) are loaded.

Asynchronous Diagnostics

Figure [3] is a software function program that can be used to test the integrity of the data bus as well as TxDA, RxDA, TxDB and RxDB. The program starts by initializing channel 'A' and 'B'. Next the MR pointers are reset and MR1A is read back and compared to the value written. If the compare passes, a relay is turned on that shorts TxDA with RxDA and TxDB with RxDB. Since the channels are in the normal mode, this will result in an external loop back. Transmitter 'A' is loaded with 256 characters as the transmitter comes ready. When the receiver interrupts the CPU, the receive FIFO is read and the contents compared with what had been transmitted. If all 256 characters are received correctly, channel 'B' is tested in the same manner. Since the SCC2691 does not have A3 to select channel 'B', the second half of this test is a simple retest of the SCC2691 using different initialization values.

SCN2681/68681 UNIQUE FUNCTIONS

Delta Break Anomaly

When the 'Rev E' parts are powered-on, some have the delta break bits set (ISR[6:2]). This can cause two possible problems: (1) if these bits are masked on in the IMR, they will immediately get a break interrupt; (2) the first characters received into the RHR will be flagged with errors (framing, break or parity) in SRA/B even though the characters were received correctly.

The way to clear these errors is to issue an external hardware reset a second time after the power-up reset, or to issue a clear delta break command to CRA or CRB before enabling the transmitter and receiver. The best method is to first, disable the transmitter and receiver; second, initialize all registers (Mr1, MR2, CSRA/B, ACR, etc.), and then clear all errors through the command registers. To be effective, this must take place at the end of the initialization routine. The ending string of commands to CRA or CRB might look like:

CRX = 50H (clear delta break bit)
CRX = 20H (reset receiver)
CRX = 30H (reset transmitter)
CRX = 45H (clear errors, enable Tx/Rx)

RTS/CTS Functions

When using the RTS/CTS functions, care must be taken to follow the flow chart in the data sheet on how to set up RTS. Although the RTS output will negate automatically, the output must first be asserted by writing a '1' to the appropriate output pin after the transmitter has been enabled, and before the first byte of the message is loaded into the THR. When the receiver controls negation of RTS, a '1' must be written to the appropriate output pin immediately after enabling the receiver.

When the receiver is controlling the negation of RTS, the sending transmitter will be stopped when the FIFO is full and the start bit of a fourth character is detected in the RSR. If the sending transmitter is a Philips Semiconductors part, the transmission will be ended when the character currently in the TSR is finished being shifted out on TxD.

When CTS goes high, the transmitter clock is stopped after the current character is shifted out on TxD. The only problem this causes is that the TxEMPT bit will not set in the SRA or SRB (even if the transmitter is empty), until the clock starts running again (when CTS goes back low).

The receivers are designed to hold three characters in the RHR and one character in the RSR. If the sending transmitter is not a Philips Semiconductors part, the character in the RSR may be overrun by a fifth character. For example, if the sending transmitter is made by Intel, the transmitter will continue to empty both the THR and the TSR when its CTS input is high. Although it will not allow any other characters to be transmitted, the receiver shift register (RSR) is still overrun.

Input Port

The 40-pin version of the SCN2681 and SCN68681 have a 7-bit input port that can be read through two different means. The port can be read in parallel by doing a read at address 'OD' hex. The lower four bits of the input port can also be read through the input port change register (IPCR). The bits in the IPCR will change as IP0–IP3 change. IPCR[0:3] show the current state of IP0–IP3. IPCR[4:7] will be set if a change of state has occurred since the IPCR was read last. Users will note that there can be differences between the data in the lower four data bits when a read is executed at address 'OD' hex. The reason for the difference is that the IPCR is updated by the internal state machine which is run on a 38.4k clock. It will take at least one clock time (25µs) to update the IPCR. Since a read of the input port is done immediately, there can be a difference in the two values.

In order to demonstrate how IP0–IP3 can interrupt the CPU, assume that the IP0 input is connected to some critical element. The interrupt is enabled by writing a one to the delta IP0 interrupt (ACR[0]) and also to the input port change mask (IMR[7]). The delta

change bits (IPCR[4:7]) should be reset by performing a read of the IPCR. When IP0 changes, IPCR[4] will be set and with it ISR[7] will set, causing an interrupt. The interrupt is reset by a read of the IPCR.

Output Port

The output port on the 40-pin version of the SCN2681 and SCN68681 has eight outputs. Each output can be set or reset by performing a write to address 'OE' or 'OF' hex ('OE' = set output port bits, 'OF' = reset output port bits). These bits are initially in the high state. Writing a one to a location at address 'OE' hex will force that individual output low. Writing a one to the same location attress 'OF' hex will reset that output to a high. For example, OP7 can be forced low by writing an '80' hex on the data bus to address 'OE' hex. Any output can be forced to toggle high and low as long as the output has not been programmed in the OPCR as a special function. If any output in the OPCR has been selected, that output will be under control of the internal state machine and the user will no longer have control of its polarity. Note that if any of the bits in OPCR[4:7] are set, the output will be considered an open drain interrupt and will need a pull-up resistor.

SCN2681 Bus Interface

The CEN and RDN, and the CEN and WRN signals are internally ANDed in the SCN2681. Because of this arrangement, the signal last asserted initiates the cycle and the signal first negated terminates the cycle. For write operations, the rising edge of either CEN or WRN latches data into the SCN2681 registers and terminates the cycle. Due to the relationship of these segnals, a number of different bus interfacing techniques can be used. Some users have grounded CEN and used an address decoder to pulse RDN and WRN. Others have pulsed CEN while driving WRN with a static R/WN and RDN with the inverse static line RN/W. Still others have used the conventional method of pulsing all three lines. In all cases the interface works.

Note 10 of the AC Electrical Characteristics states that consecutive writes to the same command register (CRA or CRB) require at least three edges of the X1 clock after the device has been deselected. This is necessary since data to the command register is only latched on the rising edge of WRN (or CEN) and three extra edges are needed for execution of the comand by the state machine.

SCN68681 Bus Interface

The SCN68681 write operation can be completed by either using CSN or the falling edge of DTACK. As with the SCN2681, there are limitations on how frequently the device can be accessed. This is given by the parameter t_{CSW}, which provides the value of the minimum high time of CSN (t_{CSW} minimum = 160ns). DTACK is a clocked output which is generated by the first two rising edges of the X1 clock after CSN has been asserted. Because CSN is asynchronous with respect to the occurrence of the rising edge of X1, the assertion of DTACK can vary as much as one full X1 clock period between device selections. Note that DTACK is an open drain output when asserted and needs a pull-up resistor (see Reading Reserved Registers for a review of possible interface problems).

MIDI INTERFACE USING THE SCC2691

The following is a good example of how the SCC2691 can be applied as a serial interface used with musical instruments (MIDI).

The musical instrument digital interface (MIDI) is setting the standard on virtually all new electronic musical instruments. Synthesizers, drum machines, sequencers, and other music related devices as well as home computer add-ons are employing this serial data interface. The digital interface operates asynchronously, at 31.25k baud, with one start bit, eight data bits, and one stop bit. The physical interface operates via an opto-isolated 5mA current loop. Multiple instruments are connected to each other in a 'daisy-chain' fashion via 'in', 'out', or 'thru' ports.

Hardware Interface and Data Format

Figure 5 shows the hardware interface used by the MIDI. Note that the MIDI 'thru' connect is an optional connection that provides a copy of the received MIDI 'in' data.

MIDI data format and baud rate:
Start bits – 1
Stop bits – 1
Data bits – 8
Parity – none
Baud rate – 31.25k

SCC2691 Baud Rate Selection

The SCC2691 on-board baud rate generator for the receiver and transmitter is from 18 fixed rates. With a given clock frequency of 3.6864MHz, via its internal divider circuitry, virtually all common baud rates encountered in low speed data communications. The following baud rates are available:

CSR[7:4]	ACR[7] = 0	ACR[7] = 1
0000	50	75
0001	110	110
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
1101	Timer	Timer
1110	MPI – 16X	MPI – 16X
1111	MPI – 1X	MPI – 1X

The MIDI baud rate, however, seems to be an exception in that an external clock of the appropriate frequency would have to be used in order to generate the required 31.25k baud rate. By changing the frequency of the external crystal, the desired baud rate can be achieved. Referring to the list of baud rates, note that the division ratio when using the 38.4k baud rate with the standard crystal frequency works out to be 96. If the crystal frequency is changed to 3MHz and using 96 as the division ration, the MIDI 31.25k baud rate can be generated internally. Also note that changing the crystal frequency to 3MHz does not violate the minimum/maximum clock specification (2 to 4MHz), therefore no problem is created in the remaining timing specifications. To implement this clock scheme, program the clock select register (CSR = CCH) and the auxiliary control register (ACR[7] = 0.

```
            BEGIN
;
;   2681 MULTIDROP OR 2691 WAKE-UP MODE TEST ROUTINE
;   CHANNEL 'A' TXD OUTPUT IS TIED TO CHANNEL 'B' RXD INPUT BY WIRE
;
;
MR1A        EQU     $7F001
MR1B        EQU     $7F011
MR2A        EQU     $7F001
MR2B        EQU     $7F011
SRA         EQU     $7F003
SRB         EQU     $7F013
CSRA        EQU     $7F003
CSRB        EQU     $7F013
CRA         EQU     $7F005
CRB         EQU     $7F015
RHRA        EQU     $7F007
RHRB        EQU     $7F017
THRA        EQU     $7F007
THRB        EQU     $7F017
ACR         EQU     $7F009
ISR         EQU     $7F00B
IMR         EQU     $7F00B
CTUA        EQU     $7F00D
CTUR        EQU     $7F00D
CTL         EQU     $7F00F
CTLR        EQU     $7F00F
RELAY       EQU     $7FC001
;
INIT:       MOVE.B  #$3A,CRA        ;RXT TXA
            MOVE.B  #$2A,CRB        ;RST RXB
            MOVE.B  #$1F,MR1A       ;MULTIDROP, 8 BITS, A/D=1
            MOVE.B  #$1B,MR1B       ;MULTIDROP, 8 BITS, A/D=0
            MOVE.B  #07,MR2A        ;NORMAL, STOP=1
            MOVE.B  #07,MR2B
            MOVE.B  #$66,CSRA       ;TXC=RXC=1200 BAUD
            MOVE.B  #$66,CSRB
            MOVE.B  SRB,D1          ;SAVE CHAN B STATUS
            MOVE.B  #06,CRA         ;ENABLE TX CHAN A
CHK0:       MOVE.B  SRA,D0          ;READ CHAN A STATUS
            BTST    #03,D0          ;IS TXEMT?
            BEQ     CHK0            ;WAIT UNTIL TXEMT=1
            MOVE.B  #$0AA,THRA      ;LOAD ADDRESS IN THRA
CHK1:       MOVE.B  SRA,D0
            BTST    #03,D0          ;IS TXEMT=1?
            BEQ     CHK1            ;WAIT UNTIL TX IS EMPTY
            MOVE.B  #$0A,CRA        ;DISABLE TX A
CHK2:       MOVE.B  SRB,D2          ;IS RXRDY=1 CHAN B?
            BTST    #0,D2           ;POLL UNTIL RXRDYB=1
            BEQ     CHK2
            MOVE.B  RHRB,D7         ;DISABLE RHR CHAN B
            CMPI.B  #$0AA,D7        ;DID CHAN B RECEIVE ADDRESS?
            BEQ     DATCHK          ;YES, CONTINUE, ELSE STOP
            TRAP    #15
DATCHK:     MOVE.B  #$09,CRB        ;ENABLE TX FOR CHAN B
            MOVE.B  #$1A,CRA        ;RESET MR POINTERS CHAN A
            MOVE.B  #$1B,MR1A       ;REWRITE MR1A (A/D=0)
            MOVE.B  #$3A,CRA        ;RESET TX A
            MOVE.B  #06,CRA         ;ENABLE TX A
            MOVE.B  #$55,THRA       ;SEND MESSAGE (DATA BYTE=55)
ENDCHK:     MOVE.B  SRB,D0
            BTST    #0,D0           ;WAIT UNTIL RXRDY=1 (CHAN B)
            BEQ     ENDCHK
            MOVE.B  RHRB,D7         ;SAVE DATA CHAR
            TRAP    #15
```

SD00665

Figure 1. SCN2681 Multipdrop or SCC2691 Wake-Up Mode

Hardware Elimination via Mode Selection

As mentioned, the MIDI specification includes an optional 'thru' connection providing a direct copy of the received data stream to other MIDI devices further down the 'daisy chain'. By utilizing the auto-echo feature of the SCC2691, the additional hardware required for the 'thru' connection can be eliminated. In auto-echo mode, the received data is reclocked and retransmitted on the TxD output using the receiver clock. Communications between receiver and CPU continues normally but the CPU-to-transmitter link is disabled. To invoke this mode of operation, set mode register 2(MR2[7:6]=01). When switching from auto-echo back to normal mode, the transmitter will remain in auto echo until a stop bit is transmitted if the deselected process occurs immediately after the receiver has sampled the stop bit and the transmitter happens to be enabled. Note that in auto-echo mode, it is not necessary to enable the transmitter.

Wake Up Mode

The SCC2691 wake up mode provides automatic wake up of the receiver through address frame recognition. In this mode, a master station transmits an address character followed by data characters for the addressed slave station. The slave stations, whose receivers are normally disabled, examine the received data stream and wake up the CPU upon receipt of an address character. The SCC2691 data sheet describes this procedure.

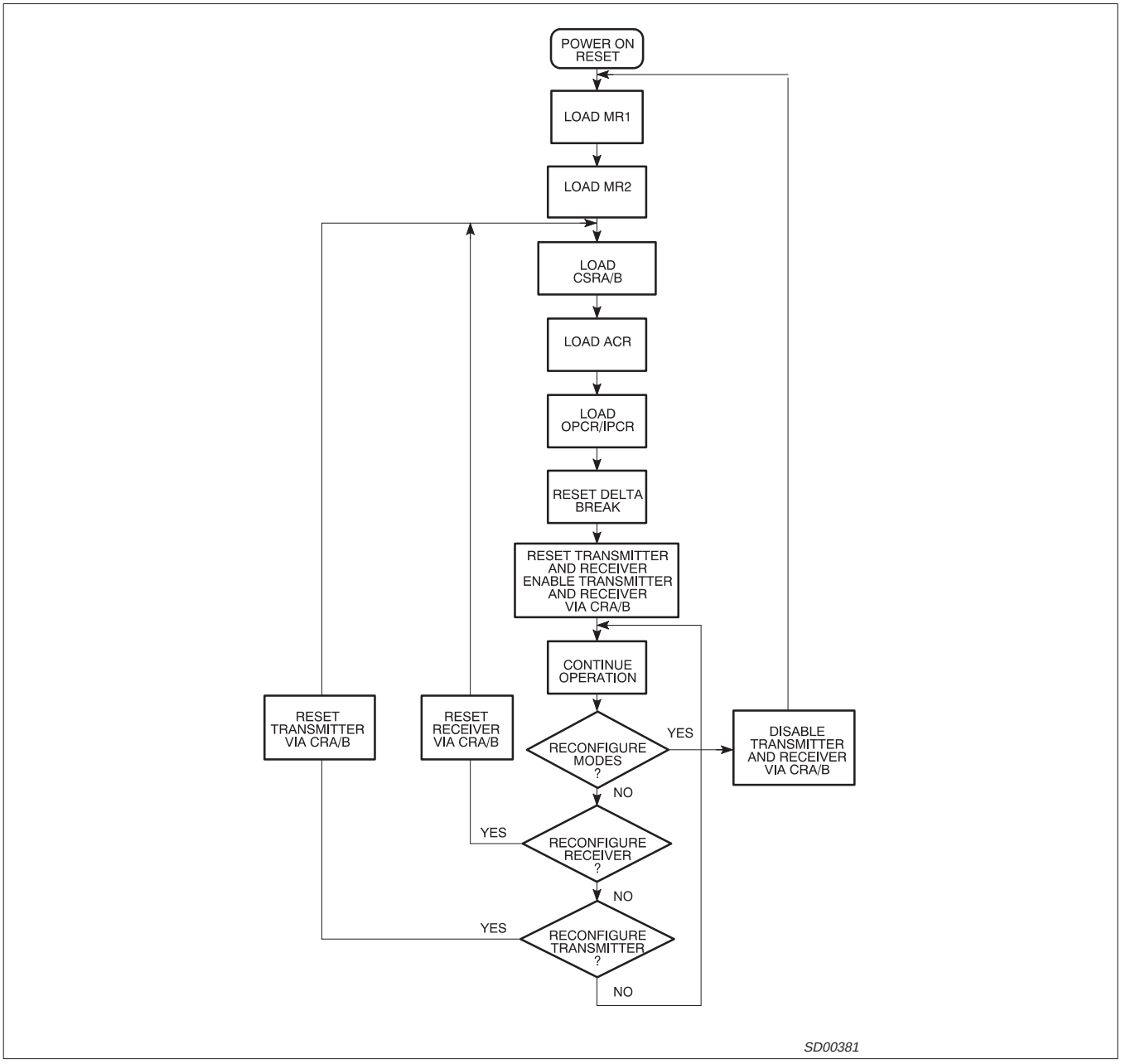

Figure 2. Asynchronous Initialization

```
;
;  2681/2691 FUNCTION PROGRAM
;    THIS PROGRAM VERIFIES THE DATA BUS IS GOOD BY READING BACK THE VALUE
;    IN MR1.  IF THIS FIRST TEST PASSES, CHANNEL 'A' AND THEN 'B' ARE TESTED
;    BY WRITING 256 CHARACTERS TO THE TRANSMITTERS (A & B) AND THEN VERIFYING
;    THE CONTENTS READ FROM THE RECEIVERS (A &B).
;
INIT:       MOVE.B    #$1A,CRA
            MOVE.B    #$30,CRA          ;RESET TX
            MOVE.B    #$20,CRA          ;RESET RX
            MOVE.B    #$13,MR1A         ;NO PARITY, 8 BITS
            MOVE.B    #7,MR2A           ;NORMAL, STOP=1
            MOVE.B    #$66,CSRA         ;TXC=RXC=1200 BAUD
            MOVE.B    #$10,CRA          ;RESET POINTER, DISABLE TX–RX
            MOVE.B    MR1A,D1           ;READ MR1A
            MOVE.B    #$13,D0
            CMP.B     D0,D1             ;COMPARE DATA VALUES
            BEQ       TEST1             ;IF COMPARE DATA BUS IS OK
            TRAP      #15               ;STOP IF FAIL
;
;THIS TEST SENDS CHAR FF THRU 00 THRA THEN READS THEM BACK.
;RX DATA IS COMPARED WITH TX DATA FROM CHANNEL 'A' ONLY.
;NOTE: RELAY SHORTS TX TO RX FOR NEXT TWO TESTS.
;
TEST1:      MOVE.B    #1,RELAY          ;SHORT TX TO RX AND CTS TO RTS
            MOVE.B    #$50,CRA          ;RESET DELTA BREAK
            MOVE.B    #$20,CRA          ;RESET RECEIVER
            MOVE.B    #$30,CRA          ;RESET TRANSMITTER
            MOVE.B    #$45,CRA          ;CLR ERRORS, ENABLE TX–RX
            MOVE.W    #$100,D7          ;SETUP FIRST SEND CHAR
TEST1A:     SUBI.B    #1,D7             ;DEC D7 UNTIL D7=0
            BEQ       TEST2             ;IF D7=0 GO TO NEXT TEST
            MOVE.B    D7,THRA           ;SEND NEXT CHAR TO TRANSMITTER
WAIT1:      BTST      #0,SRA            ;IS RECEIVER READY?
            BEQ       WAIT1
            MOVE.B    RHRA,D1           ;FETCH RECEIVED CHAR
            CMP.B     D7,D1             ;IS SENT CHAR=RECEIVED CHAR
            BEQ       TEST1A            ;IF SO REPEAT OPERATION
            TRAP      #15               ;STOP IF FAIL
;
;THIS TEST CHECKS CHAN 'B' IN THE SAME WAY AS CHAN 'A' IN TEST1.
;ONLY THE DATA FORMAT HAS BEEN CHANGED.  (SINCE THE 2691 HAS NO ADDRESS
;A2, THIS TEST WILL SIMPLY BE A RETEST OF THE SERIAL CHANNEL.)
;
TEST2:      MOVE.B    #$1A,CRB          ;DISABLE TX–RX, RESET MR PNTR
            MOVE.B    #7,MRB1           ;ODD PARITY, 8 BITS
            MOVE.B    #$0F,MR2B         ;NORMAL, TWO STOP BITS
            MOVE.B    #$0BB,CSRB        ;9600 BAUD
            MOVE.B    #$50,CRB          ;RESET DELTA BREAK
            MOVE.B    #$30,CRB          ;RESET TX
            MOVE.B    #$20,CRB          ;RESET RX
            MOVE.B    #$45,CRB          ;CLR ERRORS, ENABLE TX & RX
            MOVE.W    #$100,D7
TEST2A:     SUBI.B    #1,D7             ;DECREMENT D7
            BEQ       STOPIT            ;IF D7=0 STOP SENDING CHAR
            MOVE.B    D7,THRB           ;IF D7>0 WRITE TX HOLDING REG
WAIT2:      BTST      #0,SRB            ;TEST FOR RX READY
            BEQ       WAIT2             ;IF NOT READY, LOOP
            MOVE.B    RHRB,D1           ;PICK UP RECEIVED CHAR
            CMP.B     D7,D1             ;COMPARE TX CHAR WITH RX CHAR
            BEQ       TEST2A            ;IF THEY COMPARE, SEND AGAIN
STOPIT:     TRAP      #15               ;END ROUTINE
            END       INIT
```

SD00666

Figure 3. SCN2681/SCC2691 Function Program

For MIDI applications, this mode functions in the following manner. MIDI data types are divided into status bytes and data bytes. The MSB (D7) of the received data determines whether a status byte (D7 = 1) or a data byte (D7 = 0) has been received. By programming the receiver for seven data bits versus the transmitted eight data bits, the 8th received bit can be interpreted as the address/data (A/D) bit, thereby indicating a status byte transfer or data byte transfer.

If the receiver is disabled, a status byte transfer (A/D = D7 = 1) will set the RxRDY status bit and load the character in the receive register FIFO. This RxRDY status can be programmed to appear on the multi-purpose output pin thereby 'waking up' the CPU. Depending on the status message, the CPU can then either ignore (Rx disabled) or respond (Rx enabled) to data bytes which might follow.

For example, if the MIDI assigned channel is 5, a status byte generated by the MIDI master to channel 5 (D0–D3, a voice message perhaps), may appear as shown in Figure 4. With the A/D bit set, the receiver interprets an address (status), loads data into the RHR FIFO, and sets the RxRDY in the status register (SR). This RxRDY bit can perform the wake up function to the CPU. The CPU can then examine the data (status byte) to determine if it is the receiver (channel) being addressed. If so, as in the above example, the CPU would then enable the receiver to accept the incoming data byte(s). Once the incoming data bytes have been received, the CPU can then disable the receiver and continue handling other functions until the receiver again indicates a status has been received.

System Interconnect and Other Considerations

The bus circuitry of the SCC2691 is flexible enough to allow easy interfacing to virtually any microprocessor or microcontroller. The CE and RDN/WRN lines are ORed internal allowing either the RDN, WRN, or CE lines to initiate a data transfer. The signal asserted last initiates the cycle and the signal deasserted first terminates the cycle. Figure 5 illustrates an all CMOS MIDI interface (other than the opto-coupler), using an 80C49, HC logic and the SCC2691. This circuit draws less than 50mA at full speed and less than 1mA in standby mode. Board space reduction can also be attained by using Philips SMD packaging, thus providing a compact, low-power MIDI interface.

Transmitted byte (status byte, voice message on channel 5)

Start Bit	LSB D0	D1	D2	D3	D4	D5	D6	MSB D7	Stop Bit
1	1	0	1	0	0	0	0	1	1

Transmitted byte (status byte, voice message on channel 5)

Start Bit	LSB D0	D1	D2	D3	D4	D5	MSB D6	A/D	Stop Bit
1	1	0	1	0	0	0	0	1	1

SD00664

Figure 4. Status Byte Generated by MIDI Master to Channel 5

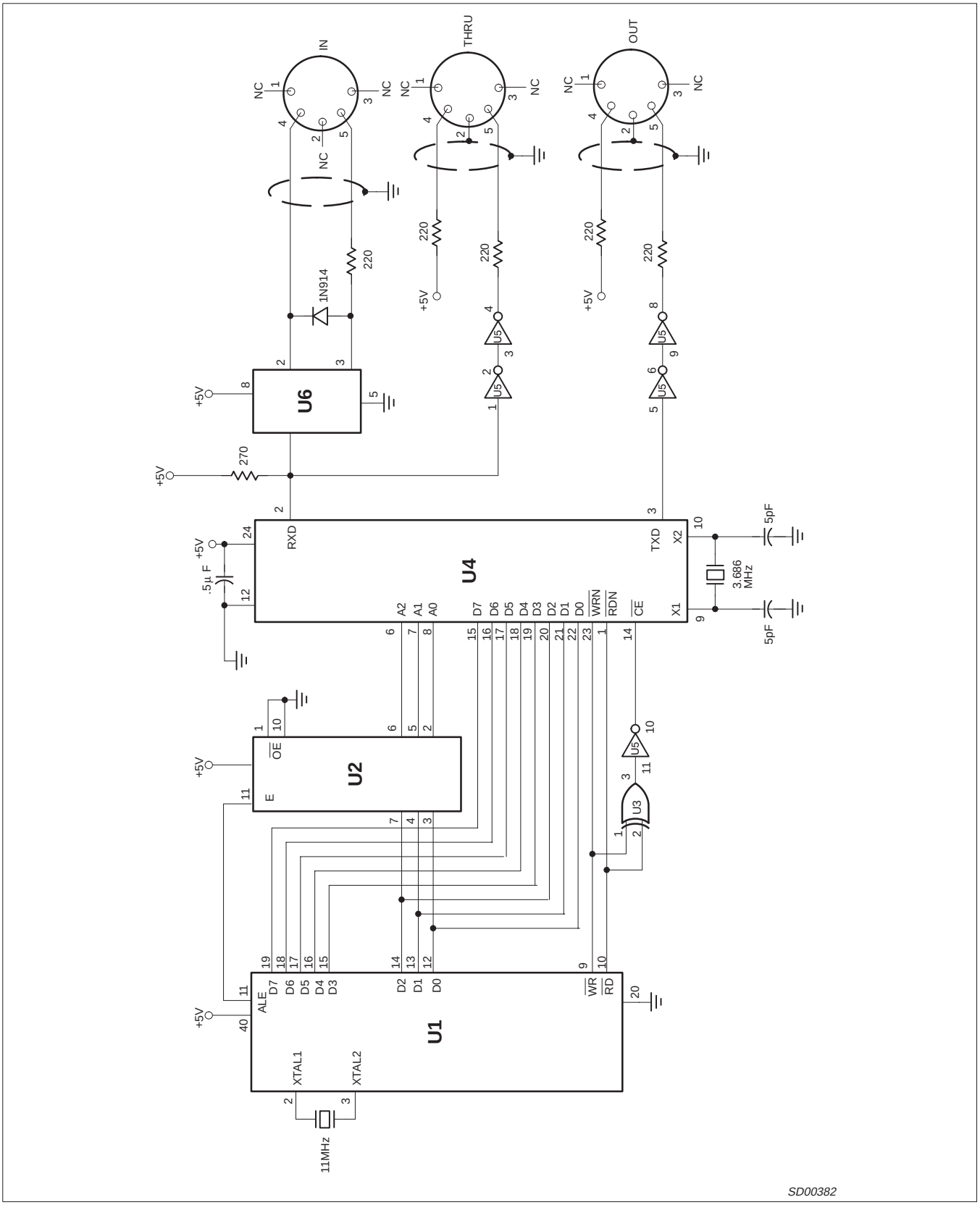

Figure 5. MIDI Interface Using SCC2691

NOTES

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

Date of release: 09–98

Document order number: 9397 750 04566

Let's make things better.

Philips Semiconductors

PHILIPS