MODEL 242
OPERATION MANUAL

SEQUENTIAL

SEQUENTIAL
Publications Department

Publications Number: CM242A
Issued: October, 1984

MODEL 242

MIDI INTERFACE CARTRIDGE

FOR THE COMMODORE 64[R]

OPERATION MANUAL

By Stanley Jungleib

and Chet Wood

Sequential
POSTBUS 16
3640 AA Mijdrecht
Netherlands
02979-6211
TELEX: 12721 SQNTL NL

Sequential
3051 North First Street
San Jose, CA 95134-2093
U.S.A.
408/946-5240
TELEX: 4997150 SEQCIR

MODEL 242

MIDI INTERFACE CARTRIDGE

FOR THE COMMODORE 64®

OPERATION MANUAL

By Stanley Jungleib

and Chet Wood

Manual No. CM242A
Issued: October, 1984

Table of Contents

OVERVIEW

Congratulations on your purchase of the Sequential Circuits Model 242 MIDI cartridge for the Commodore 64 (C64). This cartridge, together with the 900 series of diskette-based software from Sequential, allows you to perform minor miracles with your MIDI-equipped musical instruments.

To use the Model 242, you need the following:

Commodore 64 computer and power supply
Monitor (or TV)
Commodore floppy diskette drive
1 or 2 MIDI cables (depending on the application)
MIDI synthesizer
Audio cable and sound system, or headphones
Applications diskette (unless you plan to program the cartridge yourself)
Data diskette (depending on the application)
Footswitch (if desired)
Cable for drum machine (if used)

This operation manual is in three sections:

CONNECTION AND OPERATION covers basic installation. It is very short and intended for everyone.

HARDWARE describes the 242 and its theory of operation in detail, for the technically curious.

PROGRAMMING explains theory and gives examples in both BASIC and assembly language for those who wish to write their own MIDI programs.

CAUTION! Always switch power off to all equipment in use before connecting or disconnecting anything. Otherwise, serious damage may result to the C64 or cartridge.

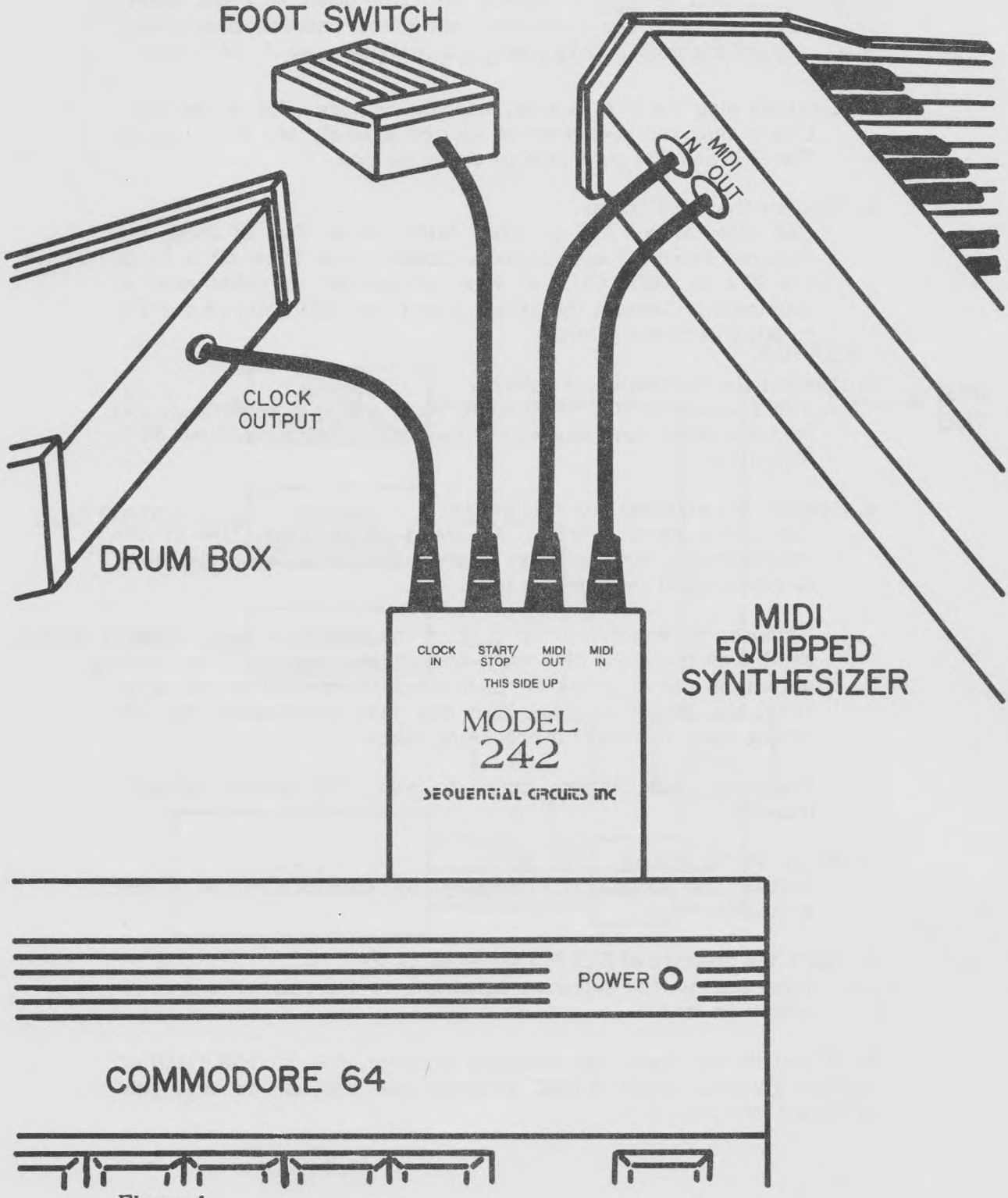


Figure 1
CONNECTION DIAGRAM

CONNECTION AND OPERATION

See figure on previous page.

1. Connect the C64 to its monitor and peripherals (diskette drive) as instructed by Commodore.

2. Switch off power to the C64, if it is not off already.

3. Carefully plug the 242, face up, into the cartridge slot of the C64.
   Check that the connector is aligned squarely and fully seated. The C64 and 242 must rest on a flat surface.

4. Connect the MIDI cables.
   Use SCI's Model 838 or other MIDI cable. Two of these are required for most applications. Connect one from MIDI IN of the 242 to MIDI OUT of your synthesizer or other musical instrument. Connect the other cable from MIDI OUT of the 242 to MIDI IN of the synth.

5. Connect the footswitch, if desired.
   The footswitch may be used to start and stop sequences, and perform other functions called for by the software. (Use SCI's Model 839.)

6. Connect the external clock, if desired.
   The clock input takes a ¼" mono phone plug. Use it for synchronizing the C64 with another sequencer, drum machine, or clock signal recorded on tape.

   Suppose, for example, that a drum machine lays down a basic rhythm on track one of a multi-track tape recorder. If the clock output of the drumbox is recorded on track two at the same time, the output of track two can then synchronize the 242 during many succesive overdubbing takes.

   For more information, refer to your 900 series software manual.

7. Switch on the system.
   Follow the order recommended by Commodore, with the synthesizer last.

8. Insert the diskette of SCI 900 series software
   Load and run the software according to the instructions in its operation manual.

9. If you do not have the software diskette, the PROGRAMMING section gives a simple BASIC program you can use to test your hardware (see page 12).

HARDWARE

The block diagram of the 242 is shown below. Two main ICs are used. The asynchronous communications interface adaptor (ACIA) and its associated buffer and optoisolator transfer MIDI serial data to and from the C64 data bus. The parallel interface adaptor (PIA) and associated circuitry route the footswitch and external clock signals to the C64. A more detailed discussion of the circuitry follows.

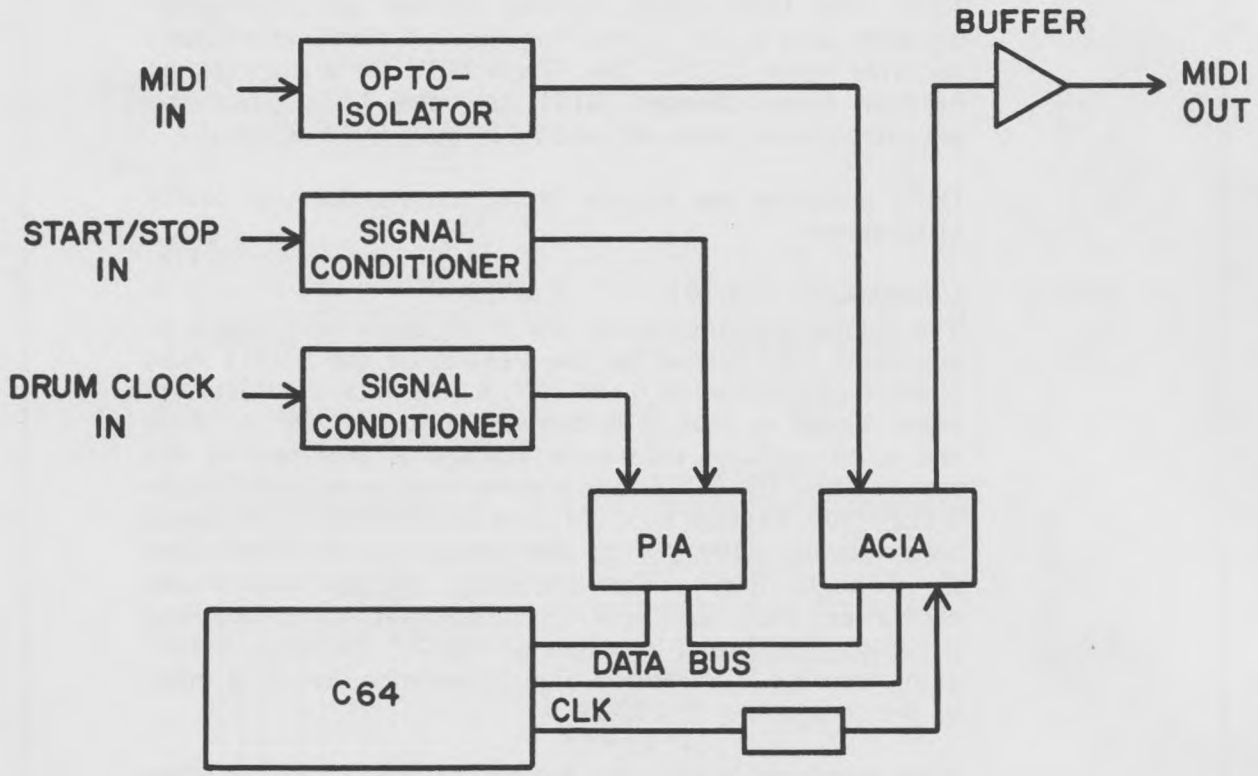


Figure 2
BLOCK DIAGRAM

The following annotated parts list describes circuit operation by part number (integrated circuits first). Refer to the schematic diagram, SD242-1, on page 9.

DESIG.	FUNCTION	SCI PART #	DESCRIPTION

INTEGRATED CIRCUITS

U101 Optoisolator I-330 Sharp PC-900

The optoisolator electrically isolates the external MIDI device and the C64/242 system. This is useful for preventing ground loops, which might cause audio hum and noise.

When data at pin 5 of MIDI IN is low (0), current flows over pin 4 and through limiter R101 and U101's internal LED. The LED lights, turning on the phototransistor between pins 4 and 5, enabling current flow, which pulls receiver input U105-2 low. When MIDI IN is high (1), no current flows through R101 and the LED. Thus the phototransistor turns off, and R114 pulls U105-2 high.

D101 protects the circuit from non-standard or faulty MIDI drivers.

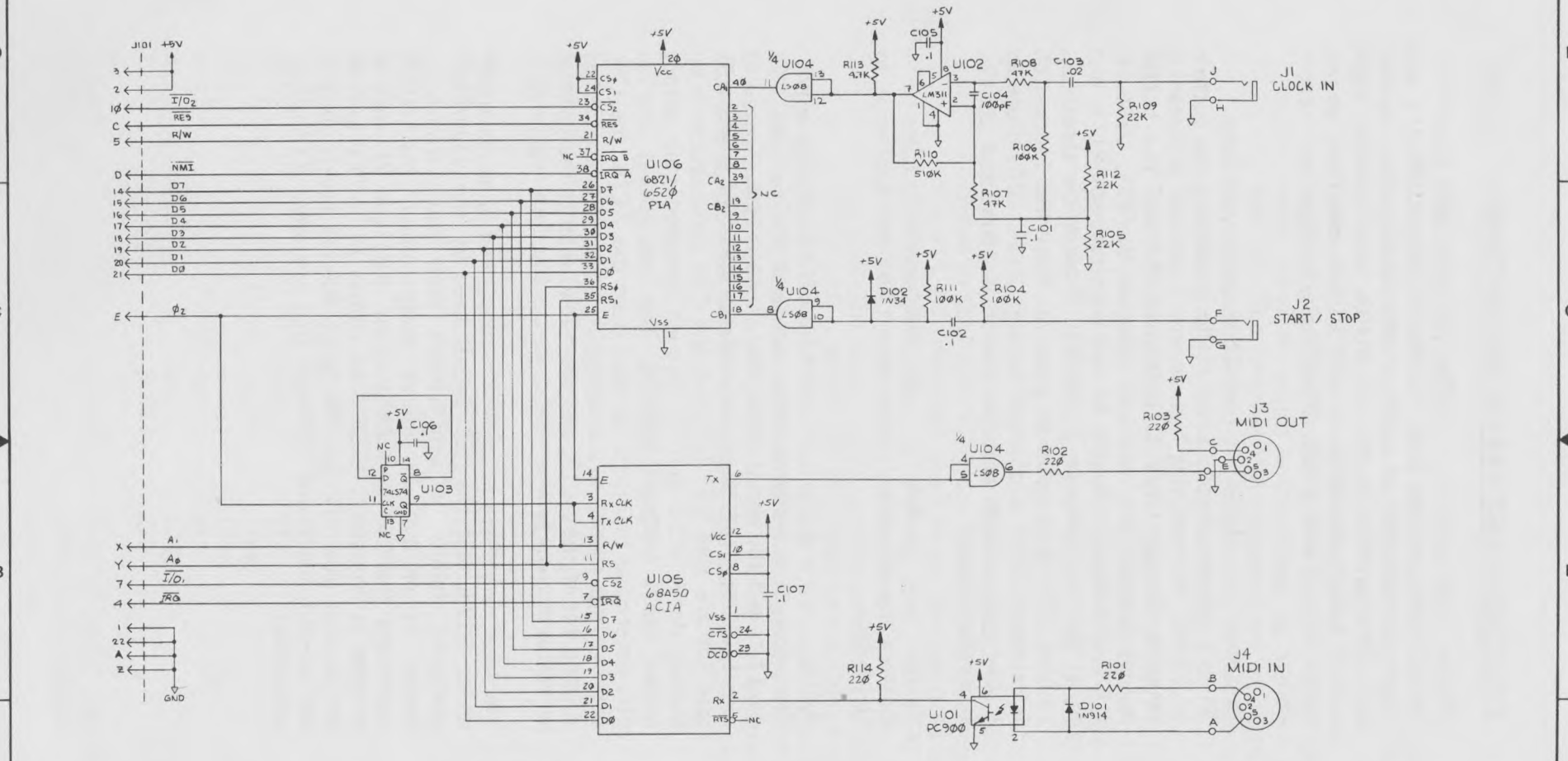
U102 Comparator I-301 LM311

The comparator squares-up the drum clock input signal to provide a logic signal for the PIA. Since the LM311 must operate on supplies of 0 and +5V, a bias network shifts the input signal so that it is centered between the supplies, and a comparator reference voltage is provided of the same value. The 2.5-V bias is generated by voltage divider R112/R105, filtered by C101, and added to both the signal path through R106 and to the comparator reference (pin 2) through R107. The incoming signal, meanwhile, encounters load resistor R109, passes through dc-blocking capacitor C103, and is offset by the 2.5 Vdc bias, before going through R108 to the high-impedance inverting input of the comparator (U102-3).

When the input is positive, output U102-7 is 0V, and when the input is negative, the output is +5V. R113 is a pull-up for the open collector output of the LM311. R110 provides positive feedback for hysteresis to prevent oscillations on slow input edges. C104 is for comparator stability.

U103-9 "T" Flip Flop I-109 74LS74

Divides the approximately 1 MHz C64 clock by two, supplying U105-3 and U105-4 with the 500 KHz from which the 31.25 kBaud data rate is derived (internally).



LAST USED	DELETED
C107	
D102	
R114	
U106	
J101	
J4	

SPARES

U104 — LS08

U103 — 74LS74

REV	DESCRIPTION	DATE	APPR
A	REL. PER ECO 0-128	8/28/84	RD

REVISIONS

APPROVALS	DATE
DWN A	6/1/84
CHECKED R. Dickinson	6/12/84
ELEC ENGR	7/1/13
MECH ENGR	
ENGR MGR	

DO NOT SCALE DRAWING

SEQUENTIAL CIRCUITS INC
3051 N. FIRST ST.
SAN JOSE, CA. 95134

TITLE: SCHEMATIC DIAGRAM C-64

SIZE: D | DRAWING NO.: SD242-1 | REV: A

SCALE: 1/1 | SHEET 1 OF 1

DESIG.	FUNCTION	SCI PART #	DESCRIPTION

U104-6 Buffer I-104 74LS08 Quad AND Gate
Serial data output from U105-6 drives U104-4/5 and appears uninverted at U104-6. When data is low, current will flow from +5V through R103, the external MIDI circuit (which should be similar to that described above, under U101,) and R102. When data is high, no current flows.

U104-8 One-shot I-104 74LS08 Quad AND Gate
When a footswitch contact is made (grounding the bottom of R104), the resulting high-to-low transition is instantly coupled through C102 to buffer U104-9/10. The buffer output triggers the interrupt input of the PIA, U106-18. C102 immediately begins to recharge through R111, and in a few milliseconds, U104-9/10 crosses its threshold, causing U104-8 to again go positive. When the footswitch is released, C102 is charged through R104. D102 protects U104 if an active voltage source is connected to the footswitch input.

U104-11 Buffer I-104 74LS08 Quad AND Gate
Buffers the output of comparator U102 and applies it to U106-40.

U105 ACIA I-066 68A50 ACIA (Motorola only)
Transfers data bidirectionally between the parallel C64 data bus and the serial MIDI. The data bus is connected to pins 15-22 of U105. Since chip selects CS0 (pin 8) and CS1 (pin 10) are tied high (active), the C64 communicates by pulling down -CS2 (pin 9) through decoded chip select output -I/O1 (address $DEXX). The two lowest address bits determine function. To read from the ACIA, the C64 sets A1 high and to write, brings A1 low. A0 is applied to RS, pin 11, of the ACIA, and when low, selects the control or status registers; when high, selects the data registers.

The baud rate clock is applied to pins 3 and 4, as discussed under U103, and the "E" signal, used by the 6500/6800 processor family to synchronize their data transfers, comes in on pin 14. If so programmed, the ACIA interrupts the C64 by pulling down -IRQ (pin 7) when a process is complete. MIDI data comes in on 'Rx' (pin 2) and is sent over 'Tx' (pin 6). Since RS-232 protocols are not needed, -DCD and -CTS are tied low, and -RTS is not connected.

DESIG.	FUNCTION	SCI PART #	DESCRIPTION
U106	PIA	I-064	6520 PIA

The PIA transfers data from the Start/Stop footswitch and drum Clock inputs to the C64 data bus. Pins 26-33 connect to the data bus, CS0 and CS1 are tied high (active), and the C64 communicates with the PIA by addressing $DFXX, which puts a low on -CS2 (-I/O2). The mode (read or write) is controlled by the C64 R/W line (pin 21). The internal PIA registers are selected by C64 address lines A0 and A1, connected to PIA functions RS0 (pin 36), and RS1 (pin 35), respectively.

The C64 reset signal, -RES, applied to pin 34, resets the PIA at power up, and the 'E' signal (described under U105) comes in on pin 25. Parallel ports A and B are not used (pins 2-17), nor are interrupt ports CA2 (pin 39) and CB2 (pin 19.)

The squared and inverted clock signal from the drumbox comes in CA1 (pin 40--from U104-11). If so programmed, the PIA will interrupt the C64 on the specified transition of the clock input by pulling down -IRQ A, pin 38, connected to the non-maskable interrupt input (-NMI), of the C64. The clock transition also sets a bit in an internal PIA register, which can be polled by the C64 if interrupts are not enabled. The Start/Stop pulse, coming in CB1 (pin 18) sets a bit in another internal register, which can also be polled. This input, however, can not be programmed to interrupt the C64, since -IRQ B (pin 37) is not connected.

OTHER PARTS

C101-3	C-045	.1 uF 50V
C104	C-004	100 pF 50V 10%
C105-7	C-045	.1 uF 50V
D101	D-005	1N914
D102	D-008	1N34
R101-3	R-003	220
R104	R-025	100K
R105	R-040	22K
R106	R-025	100K
R107-8	R-018	47K
R109	R-040	22K
R110	R-074	510K
R111	R-025	100K
R112	R-040	22K
R113	R-011	4.7K
R114	R-003	220
J101/2	EJ-006	PHONE JACK CABLE
J103/4	EJ-005	MIDI JACK CABLE

PROGRAMMING

This section gives an overview of programming the 242, followed by examples in BASIC and in assembly language. Using the Model 242 is simply a matter of reading and writing to (in BASIC, PEEKing and POKEing) the several on-chip registers of the 6520 PIA and the 6850 ACIA. It should be noted that BASIC, because of its slowness cannot be used for processing MIDI data received from the synth (a MIDI byte takes only 320 microseconds). BASIC can send data to the synth--such as musical notes at a very slow tempo, or program dumps, if you don't mind them taking a long time.

Interrupts

Not only will most applications have to use assembly language for speed considerations, but interrupts will have to be used as well. The alternative is polling at least as often as you expect an event to happen (every 320 microseconds for MIDI receiving). Drum clock inputs can also happen pretty fast--every few milliseconds. Polling so often costs a lot in execution time and code space.

Interrupts can benefit MIDI transmitting as well. The alternative, polling, could here take several different forms. For example, waiting until the last byte has been sent before sending the next one, or going off to another task while checking the ACIA every 320 microseconds (or whenever, if you're not too concerned with transmit speed) to see if it is ready to send another byte.

The ACIA interrupt is tied to the maskable interrupt of the C64. It is beyond the scope of this manual to describe C64 interrupt-handling in detail, but suffice it to say that the ACIA handler must check bits 0 and 1 to determine whether the receiver or transmitter is interrupting (see next page). If both, the receiver should be handled first.

A technique that is useful for speeding up the interrupt handlers is first-in-first-out (FIFO) buffering. In receive, the handler takes the byte from the ACIA, quickly puts it into the FIFO and returns. A background routine can then unload the FIFO when convenient. In transmit, the background process that wants to send bytes first checks that the ACIA is ready to transmit a new byte (see page 13). When it is, the first byte is placed in the ACIA transmit data register, the remaining bytes are put in the FIFO, and the transmit interrupts are enabled. The transmit interrupt handler, then, takes the next byte out of the FIFO and loads it into the ACIA. When it takes the last byte, it disables transmit interrupts. Interrupts are enabled and disabled by loading the Control register of the ACIA with the byte found in Table 2.

The PIA interrupt request is connected to the non-maskable interrupt input of the C64. PIA interrupt enable is selected in conjunction with the drum clock input polarity (see next page).

Drum Clock Input Options

If you are going to use an external drum clock, you must choose whether to trigger on the positive (low-to-high) or negative (high-to-low) transition of the clock. It is recommended that the positive transition be specified when using the SCI Drumtraks. Determine from Table 1 the mode byte to load into Control Register A of the PIA.

NOTE: The clock polarity given in Table 1 refers to the input to the 242 cartridge (J101), not the input to the PIA itself (U106-40).

Table 1
PIA CONTROL MODES (DRUM CLOCK INPUT)

Transition	Interrupts	Mode
Positive	Disabled	4
Positive	Enabled	5
Negative	Disabled	6
Negative	Enabled	7

Table 2
ACIA CONTROL MODES

Interrupts		Mode	
Receive	Transmit	Decimal	Hexadecimal
Disabled	Disabled	21	$15
Enabled	Disabled	149	$95
Disabled	Enabled	53	$35
Enabled	Enabled	181	$B5

Table 3
PIA ADDRESSING

		Address	
Register	Read or Write	Decimal	Hexadecimal
Clear A	Read	57088	$DF00
Control A	Read	57089	$DF01
Control A	Write	57089	$DF01
Clear B	Read	57090	$DF02
Control B	Read	57091	$DF03
Control B	Write	57091	$DF03

Table 4
ACIA ADDRESSING

		Address	
Register	Read or Write	Decimal	Hexadecimal
Control	Write	56832	$DE00
Transmit Data	Write	56833	$DE01
Status	Read	56834	$DE02
Receive Data	Read	56835	$DE03

Initialization

Both the ACIA and the PIA must be initialized at power-up. The C64 addresses for the registers referred to in the text can be found in Tables 3 and 4.

Initialize the PIA by first loading the mode byte from Table 1 into Control Register A. Next, since the footswitch hardware does not invert the signal, and interrupts cannot be used, load a mode byte of 4 into Control Register B.

Initialize the ACIA by loading its Control Register first with a 3 (master reset), then with the mode byte selected from Table 2.

Transmitting MIDI Data

First read the ACIA's Status Register to see if it has finished transmitting the last byte. If the ACIA is ready to send another byte, bit 1 will be high. If 0, you must wait. When ready, load the Transmit Data Register with the byte to be sent.

Receiving MIDI Data

As in transmitting, first check the ACIA's Status Register to see that a byte has been fully read in. If ready, bit 0 will be high. When the ACIA is ready, check bits 6, 5, and 4 of the Status Register. A high bit indicates the following error:

Bit 6 Parity error
Bit 5 Overrun error
Bit 4 Framing error

Then read the Receive Data Register. If an error was detected, the data is not valid, but the read must be done to reset the Status Register flag bits.

Reading the Footswitch

Read the PIA's Control Register B. If a footswitch event has occurred, bit 7 is high. After detecting an event, reset bit 7 low by reading from the Clear B register. (The data read is meaningless. The mere action of reading the register resets the bit.)

Reading the Drum Clock Input

If a clock transition of the specified polarity has occurred, bit 7 of Control Register A is high. This should be checked every few milliseconds when polling, or during the interrupt handler if interrupts are used. Again, when a clock event is detected, reset bit 7 by reading from the Clear A register.

BASIC

Now we will demonstrate these principles using an example in C64 BASIC. For real-time use BASIC is too slow, but it can be used for sending simple commands and data to the synthesizer. The following example can be used to completely check the 242 and the MIDI IN circuitry of the synthesizer. Unfortunately, since BASIC is so slow, the MIDI OUT circuitry of the synth cannot be checked.

```
10   REM BASIC PROGRAM TO VERIFY PERFORMANCE OF THE
     SCI MODEL 242.
20   LO=60
30   DELAY=300

50   REM ACIA CONSTANTS
60   WOK=2     :REM BIT 1 IS TX EMPTY FLAG
70   ROK=1     :REM BIT 0 IS RX BYTE READY FLAG
80   CR=56832  :REM CONTROL REGISTER
90   DW=CR+1   :REM TRANSMIT DATA REGISTER
100  SR=CR+2   :REM STATUS REGISTER
110  DR=CR+3   :REM RECEIVE DATA REGISTER

130  REM PIA CONSTANTS
140  RA=57088  :REM CLEAR (RESET) A REGISTER
150  CA=RA+1   :REM CONTROL REGISTER A
160  RB=RA+2   :REM CLEAR (RESET) B REGISTER
170  CB=RA+3   :REM CONTROL REGISTER B
180  MD=4
190  REM POSITIVE CLOCK TRANSITION AND NO INTERRUPTS
     FOR BOTH A AND B.
200  REM INITIALIZE PIA
210  POKE CA,MD:POKE CB,MD

220  REM INITIALIZE ACIA
230  POKE CR,3              :REM MASTER RESET
240  POKE CR,21             :REM SETUP ACIA FOR 1 START, 8
          DATA AND 1 STOP
250                         :REM BITS, 31.25 KBAUD DATA RATE,
255                         :REM     AND NO INTERRUPTS
260  REM 242 MIDI TEST
265  PRINT "TURN OFF SYNTH POWER, IF NOT OFF ALREADY"
268  PRINT:PRINT
270  PRINT "TO TEST THE 242, PLUG IT IN TO ITSELF:"
280  PRINT "CONNECT MIDI OUT TO MIDI IN"
290  PRINT "THEN HIT A 'T'"
300  PRINT:PRINT "TO SKIP THIS TEST, HIT ANY OTHER"
301  PRINT "CHARACTER."

310  GET A$ : IF A$="" THEN 310
320  IF A$="T" THEN GOSUB 2000
330  REM EXECUTE AN UPWARD GLISSANDO WITH EVER-
     DECREASING NOTE DURATIONS
```

```
332  PRINT:PRINT "NOW CONNECT THE 242'S MIDI OUT "
333  PRINT "TO MIDI IN OF THE SYNTH,"
334  PRINT "TURN SYNTH POWER ON,"
335  PRINT "AND WHEN THE SYNTH IS READY,"
337  PRINT "HIT ANY CHARACTER."
338  PRINT " YOU SHOULD HEAR A GLISSANDO."
339  GET A$ : IF A$="" THEN 339
340  WAIT SR,WOK          :REM TRANSMIT READY?
350  POKE DW,144          :REM IF SO, THEN SEND MIDI NOTE
       ON
360  REM USE RUNNING STATUS AND VELOCITY BYTE CODING
     OF NOTE ON OR OFF.

380  DD=DELAY
390  FOR NN=LO TO LO+36
400  WAIT SR,WOK          :REM TRANSMIT READY?
410  POKE DW,NN           :REM SEND OUT NOTE NUMBER
420  WAIT SR,WOK          :REM TRANSMIT READY?
430  POKE DW,64           :REM  NON-ZERO  VELOCITY  FOR
       NOTE ON
440  GOSUB 1000           :REM DELAY FOR NOTE SUSTAIN
450  WAIT SR,WOK          :REM TRANSMIT READY?
460  POKE DW,NN           :REM NOTE NUMBER AGAIN
470  WAIT SR,WOK          :REM TRANSMIT READY?
480  POKE DW,0            :REM ZERO VELOCITY FOR NOTE OFF
490  NEXT

500  REM NOW A SIMILAR DOWNWARD GLISS
510  DD=DELAY
520  FOR NN=LO+36 TO LO STEP -1
530  WAIT SR,WOK          :REM TRANSMIT READY?
540  POKE DW,NN           :REM SEND OUT NOTE NUMBER
550  WAIT SR,WOK          :REM TRANSMIT READY?
560  POKE DW,64           :REM  NON-ZERO  VELOCITY  FOR
       NOTE ON
570  GOSUB 1000           :REM DELAY FOR NOTE SUSTAIN
580  WAIT SR,WOK          :REM TRANSMIT READY?
590  POKE DW,NN           :REM NOTE NUMBER AGAIN
600  WAIT SR,WOK          :REM TRANSMIT READY?
610  POKE DW,0            :REM ZERO VELOCITY FOR NOTE OFF
620  NEXT
630  PRINT "THANK YOU"

640  REM TEST FOOTSWITCH
650  PRINT "TO TEST THE FOOTSWITCH, STEP ON IT AND "
660  PRINT "WATCH THE SCREEN FOR THE MESSAGE "
670  PRINT "'FOOTSWITCH ACTIVATED':PRINT
680  PRINT "HIT ANY CHARACTER TO END THE TEST"
690  A=PEEK (RB)          :REM  CLEAR  INTERRUPT  FLAG
       BEFORE STARTING
700  IF (PEEK(CB)AND 128)=0 THEN 730
710  PRINT "FOOTSWITCH ACTIVATED"
720  A=PEEK (RB)          :REM CLEAR FLAG AFTER RECEIVING
730  GET A$:IF A$="" THEN 700
740  PRINT:PRINT:PRINT "THANK YOU"
```

```
760  REM CLOCK INPUT TEST
770  PRINT "CLOCK INPUT TEST"
780  PRINT "A MESSAGE WILL BE PRINTED ON THE SCREEN "
790  PRINT "WHEN A CLOCK SIGNAL IS RECEIVED."
800  PRINT : PRINT "HIT ANY KEY TO END TEST"
810  PRINT : PRINT "START DRUMBOX..."

820  A=PEEK (RA)          :REM  CLEAR  INTERRUPT  FLAG
       BEFORE STARTING
830  IF (PEEK(CA)AND 128)=0 THEN 860
840  PRINT "CLOCK SIGNAL RECEIVED"
850  A=PEEK (RA)          :REM CLEAR FLAG AFTER RECEIVING
860  GET A$:IF A$="" THEN 830
870  PRINT:PRINT:PRINT "THANK YOU"
880  PRINT:PRINT:PRINT "THANK YOU"
890  PRINT:PRINT:PRINT "THANK YOU"
900  PRINT "END"
910  END

997  REM SUBROUTINE DELAY
998  REM PRODUCES A DELAY WHICH IS SMALLER EACH TIME
     IT IS CALLED
999  REM UNLESS THE "DD" VARIABLE IS RESET EXTERNALLY

1000 FOR N=INT(DD) TO 0 STEP -1
1010 NEXT
1020 DD=DD/2↑(1/6)
1030 RETURN
1040 END

1990 REM SUBROUTINE TEST 242 MIDI HARDWARE
2000 PRINT "IF TEST NOT DONE IN 8 SECONDS,"
2001 PRINT "HIT RUN/STOP RESTORE TO STOP PROGRAM,"
2002 PRINT "THEN CHECK THE MIDI CABLE."
2003 PRINT:PRINT
2005 FOR A=0 TO 255
2010 WAIT SR,WOK          :REM WAIT FOR TX EMPTY
2020 POKE DW,A            :REM SEND THIS BYTE
2030 WAIT SR,ROK          :REM WAIT FOR RECEIVED BYTE
2035 IF (PEEK(SR)AND 112)=0 THEN 2040        :REM       CHECK
       ERROR FLAGS (BITS 4,5,&6.)
2038 PRINT "*****FLAG*****":GOTO 2050
2040 IF PEEK(DR)=A THEN 2060 :REM READ AND TEST RECEIVED
     BYTE
2050 PRINT "******ERROR******"
2055 STOP
2060 NEXT
2070 PRINT "242 MIDI HARDWARE CHECKS OK"
2080 RETURN
2090 END
```

PROGRAMMING IN ASSEMBLY LANGUAGE

The following code segments illustrate the programming principles described above. (To run them on the C64, you need an assembler and debugger.)

1. To initialize the ACIA for MIDI:

```
        LDA #$03        ;Master reset
        STA $DE00
        LDA #$15        ;Setup for 1 start bit+8 bits+1 stop bit
        STA $DE00       ; and 31.25 kBaud
```

2. To send MIDI OUT:

```
LOOP    LDA $DE02       ;Read ACIA status register
        AND #$02        ;Check transmit empty flag
        BEQ LOOP        ;Loop until empty
        LDA MIDIBYTE    ;Load in byte to send
        STA $DE01       ;Send MIDI OUT
```

3. To receive MIDI IN:

```
LOOP    LDA $DE02       ;Read ACIA status register
        LSR             ;Check receive full flag
        BCC LOOP        ;Loop until ready
        LDA $DE03       ;Read MIDI IN
```

4. To initialize the PIA:

```
        LDA MODE        ;Use control registers to set up
        STA $DF01       ;operating mode.
        STA $DF03       ;Obtain mode from Table 1
```

5. To read drum clock input:

```
        LDA $DF01       ;Read control register
        BPL EXIT        ;Branch if no drum clock yet
                        ;Falls thru if interrupted by drum clock
        LDA $DF00       ;Clear the interrupt flag bit
```

6. To read footswitch input:

```
        LDA $DF03       ;Read control register
        BPL EXIT        ;Branch if no footswitch press
        LDA $DF02       ;Clear interrupt flag
```