Improved KIM Communication Capabilities

Add new I/O capabilities to your KIM with this software/hardware combination.

Ralph Tenny
P.O. Box 545
Richardson, Texas 75080

Code and Text Transfer

Unless he has a programmer, the small system owner often wonders how to program EPROMs for his system. Or, if he locates a friend with a programmer on his system, he then must figure out how to develop the program code on the KIM, test it, and then get the code into the system with the programmer. It is extremely likely that any scheme involving re-entry of the code in the second system will introduce errors, so it is desirable that the KIM produce a copy of its own code in a form usable by the second system.

First you need a program which puts out the exact memory image of the developed and debugged program. KIMOUT is such a program, which uses a second RS-232 port added to KIM. The reason that KIM's serial port is not suitable (in many cases) is that the KIM port has a hardware echo built in. Also, in some cases, the I/O lines driving KIM's serial port are disturbed by the operating system. Thus, a second port (described later) allows you to have an unrestricted and undisturbed, echo-free serial I/O port which won't ruffle the feathers of any other computer system it may be talking to.

The chief difference between KIMOUT and any other memory dump program is that KIMOUT does no data formatting, and inserts no characters which are not part of the memory image desired in EPROM. The software shown uses the second serial I/O program which was adapted from KIM's software to drive the second serial port. All the "new" software is part of an additional 2K of EPROM added to KIM and located at $C000_{16}$ through $C7FF_{16}$. However, these routines have been located beginning at 0200 and 0300 by making the appropriate changes in addresses.

Once the program to be ROMmed is ready, KIMOUT is given the starting and ending addresses of the program as follows:

	Start	End
Address Low	0002	0004
Address High	0003	0005

Table 1

Baud Rate	17F2 CNTL 30	17F3 CNTH 30
110		
300	E8	00
1200	35	00

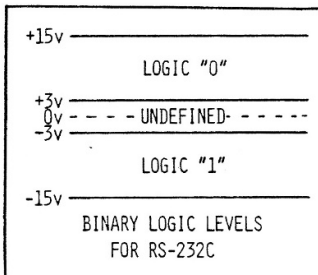

Figure 1: **RS-232 signals have a voltage "deadband" between +3V and −3V to increase the noise immunity of the equipment.**

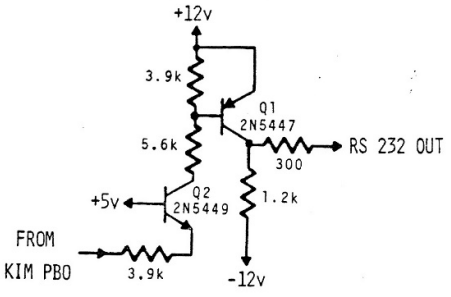

Figure 2: **Two transistors and some resistors make a very simple RS-232 output port to supplement KIM's current loop serial port.**

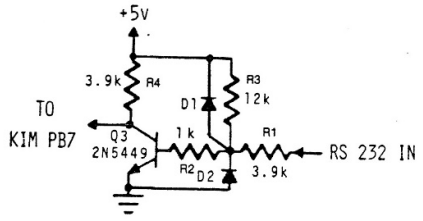

Figure 3: **One transistor buffered by resistors and diodes makes an RS-232 input port with protection against unexpected voltages.**

Set the timing constants in CNTL30 and CNTH30 (17F2, 17F3) for the proper data rate (see table 1), connect the two computers, start the receiving program in the other computer, then start KIMOUT. When KIMOUT has finished, it will re-light the KIM display, and you can terminate the receiving program.

In my case, the receiving program was in a TM990/189 (TI's University Board), which uses only 300 baud. Once the data has been transferred, I check starting and ending bytes, and a few representative other locations in the '189 memory, then dump the data to audio tape. (The TM990/189 will make a digital tape if a Model 733 TI terminal is available.) The '189 at work can read this audio tape and there is a programmer attached to it. About five minutes after dumping the tape, I have another EPROM for KIM!

It should be noted that some EPROM programmers (and some computers) will require that data handled in this manner be formatted into blocks with checksums. The tapes themselves use TI's tag loader format, so the actual transfer between the two University Boards is protected by checksums. So far, I have never encountered an error introduced by the process described, so maybe I've been lucky!

The program called TRANSLATE contains three smaller programs which cooperate in another type of data transfer. The Radio Shack TRS-80C™ computer has a 600 baud printer port, and the software issues only carriage returns instead of the CRLF pair issued by KIM and many other computers at the end of a line. I had no access to any 600 baud printers, and even my CRT terminal needed the line feed to present a picture of the TRS-80C output. So, the first section of TRANSLATE (SETUP) beginning at 0200 will read code or text from memory and add a line feed to any carriage return found.

The second section of TRANSLATE (RCV) beginning at 0238 will receive any continuous string of ASCII characters and place the characters in contiguous memory locations as long as there is memory left. If the string over-writes the end of the buffer (on KIM, the available buffer is 03E0-13FF), it quits listening and bounces back to the KIM monitor. Finally, the third section of TRANSLATE (CLEAR) clears memory beginning at the address specified in 0002 and 0003 (the same buffer is used for all sections of TRANSLATE) and extending through 13FF.

```
0800              ;**************************
0800              ;*                        *
0800              ;* COMMUNICATIONS SUPPORT *
0800              ;*                        *
0800              ;*     BY RALPH TENNY     *
0800              ;*                        *
0800              ;**************************
0800              ;*
0800              PBD    EQU $1702
0800              PBDD   EQU $1703
0800              CNTL30 EQU $17F2
0800              CNTH30 EQU $17F3
0800              TIMH   EQU $17F4
0800              START  EQU $1C4F
0800              CRLF   EQU $1E2F
0800              INITS  EQU $1E88
0800              OUTCH  EQU $1EA0
0800              PACK   EQU $1FAC
0800              TOP    EQU $1FD5
0800              ;
0800              SAL    EPZ $02
0800              SAH    EPZ $03
0800              EAL    EPZ $04
0800              EAH    EPZ $05
0800              YTMP   EPZ $20
0800              TMPY   EPZ $EE
0800              INL    EPZ $F8
0800              TEMP   EPZ $FC
0800              TMPX   EPZ $FD
0800              CHAR   EPZ $FE
0800              ;
0200                     ORG $200
0200                     OBJ $800
0200              ;
0200              ;***  TRANSLATE   ***
0200              ;
0200              ;THIS PROGRAM RECEIVES A HEX ASCII TEXT STRING
0200              ;OVER KIM'S STANDARD SERIAL PORT AND STORES
0200              ;THE STRING IN CONTIGUOUS MEMORY LOCATIONS.
0200              ;THIS SAME TEXT STRING CAN THEN BE OUPUT TO A
0200              ;PRINTER OR OTHER RS 232 DEVICE FOR DISPLAY.
0200              ;THIS ALLOWS KIM TO RECEIVE FROM A CPU WHICH
0200              ;HAS NO BAUD RATE SELECTION, AND TO OUTPUT
0200              ;TO A PRINTER AT ANY BAUD RATE DESIRED.
0200              ;
0200              ;THIS SECTION READS MEMORY, RESETS THE PRINTER
0200              ;(CARRIAGE RETURN-LINE FEED) IF THE CHARACTER
0200              ;IS A CARRIAGE RETURN ($0D), AND OUTPUTS
0200              ;ALL OTHER CHARACTERS.
0200              ;
0200 20881E       SETUP  JSR INITS          ;SET UP KIM STANDARD PORTS
0203 A000         INDX   LDY #$00           ;INITIALIZE Y INDEX
0205 8420                STY YTMP           ;AND POINTER REGISTER
0207 A420         OUT    LDY YTMP           ;PICK UP POINTER VALUE
0209 B102                LDA (SAL),Y        ; AND INDEX INTO TEXT BUFFER.
020B C90D                CMP #$0D           ;IS IT A CARRIAGE RETURN?
020D F011                BEQ RESET          ;IF SO, RESET THE PRINTER.
020F 20A01E              JSR OUTCH          ;OTHERWISE, OUTPUT CHARACTER.
0212 E620                INC YTMP           ;THEN BUMP THE POINTER.
0214 D007                BNE MORE           ;TEST FOR END-OF-MEMORY PAGE.
0216 18                  CLC                ;IF SO, PREPARE TO ADD
0217 A503                LDA SAH            ;GET PAGE POINTER
0219 6901                ADC #$01           ;AND INCREMENT IT
021B 8503                STA SAH            ;RESTORE PAGE POINTER
021D 4C0702       MORE   JMP OUT            ;AND KEEP TRUCKIN'
0220 202F1E       RESET  JSR CRLF           ;RESET THE PRINTER
0223 A520                LDA YTMP           ;GET THE POINTER
0225 38                  SEC                ;FORCE A CARRY
0226 6502                ADC SAL            ;TO BUMPT LO BYTE OF ADDRESS
0228 8502                STA SAL            ;AND RESTORE ADDRESS
022A A503                LDA SAH            ;GET THE HI BYTE
022C 6900                ADC #$00           ;ADD IN POSSIBLE CARRY
022E 8503                STA SAH            ;AND PUT HI BYTE BACK
0230 C914                CMP #$14           ;END OF MEMORY?
0232 D0CF                BNE INDX           ;IF NOT, MOVE ON OUT
0234 4C4F1C              JMP START          ;OTHERWISE, RETURN TO KIM
0237 00                  BRK
0238              ;
0238              ;THIS SECTION RECEIVES INCOMING HEX ASCII
```

```
0238                ;CHARACTERS AND STORES THEM IN MEMORY LOCATIONS
0238                ;DEFINED IN $02 AND $03.
0238                ;
0238 205103         RCV     JSR INIT            ;INITIALIZE SECOND PORT
023B A000                   LDY #$00            ;SET Y TO ZERO
023D 8420                   STY YIMP            ;ALONG WITH POINTER REGISTER
023F 201F03         IN      JSR GETCHP          ;READ SECOND PORT
0242 C902                   CMP #$02            ;VALID CHARACTER?
0244 30F9                   BMI IN              ;IF NOT, KEEP TRYING
0246 A420                   LDY YIMP            ;PUT POINTER IN Y REGISTER
0248 9102                   STA (SAL),Y         ;AND DEPOSIT THE BYTE
024A E620                   INC YIMP            ;BUMP THE POINTER,
024C D0F1                   BNE IN              ;TEST FOR MEMORY PAGE END
024E A503                   LDA SAH             ;IF SO, GET PAGE POINTER
0250 18                     CLC                 ;PREPARE FOR ADD
0251 6901                   ADC #$01            ;INCREMENT PAGE POINTER
0253 8503                   STA SAH             ;AND PUT IT BACK
0255 C914                   CMP #$14            ;TEST FOR MEMORY END
0257 D0E6                   BNE IN              ;IF NOT, GO GET MORE DATA
0259 4C4F1C                 JMP START           ;OTHERWISE, RETURN TO KIM
025C                ;
025C                ;THIS SECTION CLEARS A MEMORY BUFFER BY WRITING
025C                ;$00 IN EACH LOCATION
025C                ;
025C A000           CLEAR   LDY #$00            ;CLEAR INDEX POINTER
025E 98                     TYA                 ;AND THE ACCUMULATOR
025F 9102           WRITE   STA (SAL),Y         ;CLEAR MEMORY BUFFER
0261 E602                   INC SAL             ;BUMP THE INDEX
0263 D0FA                   BNE WRITE           ;TEST FOR MEMORY PAGE END
0265 A503                   LDA SAH             ;IF SO, GET PAGE POINTER
0267 18                     CLC                 ;PREPARE TO ADD
0268 6901                   ADC #$01            ;ONE TO PAGE POINTER
026A 8503                   STA SAH             ;AND PUT IT BACK.
026C C914                   CMP #$14            ;END OF MEMORY?
026E D0EC                   BNE CLEAR           ;IF NOT, CLEAR MORE MEMORY
0270 4C4F1C                 JMP START           ;OTHERWISE, RETURN TO KIM
0273                ;
0273                ;KIMOUT
0273                ;
0273                ;THIS PROGRAM UTILIZES A SECOND RS-232 PORT ON
0273                ;KIM TO OUTPUT A CONTINUOUS DATA STREAM
0273                ;(USUALLY TEXT OR PROGRAM DATA) TO AN EPROM
0273                ;PROGRAMMER OR PRINTER.
0273                ;
0280                        ORG SETUP+$80
0280                        OBJ $880
0280                ;
0280 205103         STRT    JSR INIT            ;SET UP POINTER STORAGE
0283 A900           ZERO    LDA #$00            ;SET INITIAL POINTER VALUE
0285 8520                   STA YIMP            ;IN A SAFE LOCATION
0287 A420           GET     LDY YIMP            ;LOAD POINTER INTO INDEX
0289 B102                   LDA (SAL),Y         ;GET A BYTE OF DATA
028B 200003                 JSR PRTBYT          ;AND OUTPUT IT
028E E620                   INC YIMP            ;BUMP THE POINTER
0290 18                     CLC                 ;PREPARE TO ADD
0291 A502                   LDA SAL             ;LO BYTE START ADDRESS
0293 6520                   ADC YIMP            ;TO THE POINTER
0295 8502                   STA SAL             ;FOR NEW START ADDRESS
0297 A503                   LDA SAH             ;GET HI BYTE
0299 6900                   ADC #$00            ;ADD IN POSSIBLE CARRY
029B 8503                   STA SAH             ;AND RESTORE HI BYTE
029D A502                   LDA SAL             ;GET LO BYTE
029F C504                   CMP EAL             ;AND COMPARE TO END LO BYTE
02A1 9008                   BCC NEXT            ;IF NOT, GO MOVE MORE DATA
02A3 A503                   LDA SAH             ;OTHERWISE, CHECK HI BYTE
02A5 C505                   CMP EAH             ;AGAINST END HI BYTE
02A7 F005                   BEQ OUTK            ;IF EQUAL,
02A9 1003                   BPL OUTK            ;OR BIGGER, STOP
02AB 4C8002         NEXT    JMP STRT            ;OTHERWISE DO MORE
02AE 4C4F1C         OUTK    JMP START           ;DONE, EXIT TO KIM
02B1 00                     BRK
02B2                ;
02B2                ;SERIAL I/O
02B2                ;
02B2                ;THIS PROGRAM IS A SLIGHTLY MODIFIED COPY OF
02B2                ;PORTIONS OF THE KIM-1 MONITOR FUNCTIONS;
02B2                ;WITH THE EXCEPTION OF INIT, THE LABELS HAVE BEEN
02B2                ;PRESERVED.  THE MODIFICATIONS ACCOMODATE THE USE
02B2                ;OF A SEPARATE RS-232 SERIAL PORT, IMPLEMENTED IN
02B2                ;CONJUNCTION WITH THE APPLICATIONS I/O PORT OF KIM.
```

(continued)

TRANSLATE has made it possible for me to "translate" the Radio Shack computer output from 600 baud to 300 baud for a borrowed printer. Both TRANSLATE and KIMOUT will handle any type of computer data, because they deal with exact memory images of the data. I can even generate text such as this on KIM and bring it to this word processor for final editing, formatting and printing on a daisy-wheel printer!

Add A Second RS-232 Port

One problem with the KIM port is that it has a hardware echo built in which is inappropriate in some applications. Also, since the software is all in ROM, it is impossible to modify. These problems may be simply solved by creating a second RS-232 port.

The 20 mA loop port on the KIM-1 can be converted to an RS-232 port by adding some transistors to shift the input/output levels to match RS-232 specifications. Figure 1 details the voltage levels which make up the RS-232 specification. Some RS-232 peripheral devices will work with a smaller voltage swing or other deviations from the spec, but to be sure, build the simple circuits shown in figures 2 and 3.

Figure 2 shows the output circuit. This port will swing to full RS-232 levels and should meet all drive requirements for almost any imaginable peripheral device. Q1 is the output switch, while Q2 is a non-inverting level converter which allows the full ± 12v RS-232 swing from Q1, without requiring an open-collector stage on the port line or the UART.

The problem of matching RS-232 input levels to another port pin is solved by the circuit shown in figure 3. A single transistor with input protection can accept ± 12v swings and convert them to a level KIM is happy with. R1, D1 and D2 form a protective network for the transistor base. Also R1 with R2 provides adequate input impedance for the incoming signal. R3 is a pull-up to hold the port's input line at a spacing level (logic 0) when there is no input signal.

The KIM provides the basic software UART routines. The routines (PRTBYT, GETCH, OUTSP, OUTCH, and CRLF), use bit PB0 of the KIM Control Port to drive the output, and incoming data is read on PA7. We can do about the same thing, using PB0 of the Application Port for an output and PB7 for input. With those pin

```
02B2                ;IN ADDITION, THE Y REGISTER OF THE 6502 HAS BEEN
02B2                ;SAVED WHERE APPROPRIATE.
02B2                ;
0300                      ORG $300
0300                      OBJ $900
0300                ;
0300 85FC           PRTBYT STA TEMP          ;SAVE ACCUMULATOR
0302 4A                    LSR               ;SHIFT OFF LOW NIBBLE
0303 4A                    LSR               ;TO ACCESS
0304 4A                    LSR               ;THE HIGH ORDER
0305 4A                    LSR               ;NIBBLE FOR OUTPUT
0306 201103                JSR HEXTA         ;CONVERT TO HEX AND OUTPUT
0309 A5FC                  LDA TEMP          ;GET OTHER HALF
030B 201103                JSR HEXTA         ;CONVERT TO HEX AND OUTPUT
030E A5FC                  LDA TEMP          ;RESTORE BYTE IN A
0310 60                    RTS               ;AND RETURN
0311 290F           HEXTA  AND #$0F          ;MASK OFF HI NIBBLE
0313 C90A                  CMP #$0A          ;TEST FOR ALPHA
0315 18                    CLC               ;PREPARE TO ADD
0316 3002                  BMI HEXTA1        ;NOT ALPHA
0318 6907                  ADC #$07          ;ALPHA, ADD MORE
031A 6930           HEXTA1 ADC #$30          ;FIX NON-ALPHA
031C 20A01E                JSR OUTCH         ;OUTPUT IT
031F 86FD           GETCHP STX TMPX          ;SAVE X REG
0321 84EE                  STY TMPY          ;AND Y REG
0323 A208                  LDX #$08          ;COUNT OF 8 BITS
0325 A901                  LDA #$01          ;MASK IN ACCUMULATOR
0327 2C0217         GET1   BIT PBD           ;TEST FOR START BIT
032A EA                    NOP
032B EA                    NOP
032C 30F9                  BMI GET1          ;KEEP TRYING
032E 209303                JSR DELAY         ;DELAY ONE BIT
0331 20AA03         GET5   JSR DEHALF        ;DELAY 1/2 BIT
0334 AD0217         GET2   LDA PBD           ;GET 8 BITS
0337 2980                  AND #$80          ;MASK OFF LOW ORDER BITS
0339 46FE                  LSR CHAR          ;SHIFT CHARACTER RIGHT
033B 05FE                  ORA CHAR          ;OR IN RECEIVED BIT
033D 85FE                  STA CHAR          ;AND RESTORE CHAR
033F 209303                JSR DELAY         ;DELAY ONE BIT TIME
0342 CA                    DEX               ;AND COUNT BIT
0343 D0EF                  BNE GET2          ;REPEAT UNTIL 8 BITS IN
0345 20AA03                JSR DEHALF        ;THEN, DELAY 1/2 BIT
0348 A4EE                  LDY TMPY          ;RETRIEVE Y
034A A6FD                  LDX TMPX          ;AND X
034C A5FE                  LDA CHAR          ;GET THE CHARACTER
034E 2A                    ROL               ;AND SHIFT OFF THE
034F 4A                    LSR               ;PARITY BIT, THEN
0350 60                    RTS               ;RETURN
0351 A201           INIT   LDX #$01          ;TURN ON ONE BIT
0353 8E0317                STX PBDD          ;IN THE USER PORT
0356 D8                    CLD               ;SET UP BINARY MODE
0357 78                    SEI               ;INHIBIT INTERRUPTS
0358 60                    RTS               ;AND RETURN
0359 A920           OUTSP  LDA #$20          ;ASCII SPACE
035B 85FE           OUTCHA STA CHAR          ;SAVE THE CHARACTER
035D 84EE                  STY TMPY          ;THE Y REG,
035F 86FD                  STX TMPX          ;AND X REG
0361 209303                JSR DELAY         ;ONE BIT DELAY
0364 AD0217                LDA PBD           ;READ THE PORT
0367 29FE                  AND #$FE          ;SET THE START BIT
0369 8D0217                STA PBD           ;OUTPUT THE BIT
036C 209303                JSR DELAY         ;WAIT ONE BIT TIME
036F A208                  LDX #$08          ;EIGHT BIT COUNT
0371 AD0217         OUT1   LDA PBD           ;GET THE OUTPUT BIT
0374 29FE                  AND #$FE          ;MASK START BIT
0376 46FE                  LSR CHAR          ;SHIFT BIT OUT OF CHAR
0378 6900                  ADC #$00          ;ADD IN CARRY BIT
037A 8D0217                STA PBD           ;AND OUTPUT IT
037D 209303                JSR DELAY         ;WAIT ONE BIT TIME
0380 CA                    DEX               ;COUNT THE BIT
0381 D0EE                  BNE OUT1          ;NOT DONE, GO BACK
0383 AD0217                LDA PBD           ;LOAD THE OUTPUT BIT
0386 0901                  ORA #$01          ;SET IT HGH
0388 8D0217                STA PBD           ;TO OUTPUT STOP BIT
038B 209303                JSR DELAY         ;AND WAIT AGAIN
038E A6FD                  LDX TMPX          ;REMEMBER X
0390 A4EE                  LDY TMPY          ;AND Y
0392 60                    RTS               ;AND RETURN
0393 ADF317         DELAY  LDA CNTH30        ;GET HI BYTE DELAY COUNT
```

```
0396 8DF417            STA TIMH          ;STUFF IT IN THE TIMER
0399 ADF217            LDA CNTL30        ;AND GET THE LO BYTE
039C 38          DE2   SEC               ;SET CARRY FOR SUBTRACT
039D E901        DE4   SBC #$01          ;DECREMENT LO BYTE
039F B003              BCS DE3           ;BRANCH IF NO BORROW
03A1 CEF417            DEC TIMH          ;DECREMENT TIMER VALUE
03A4 ACF417      DE3   LDY TIMH          ;AND STUFF IT IN Y
03A7 10F3              BPL DE2           ;RETURN IF NOT NEGATIVE
03A9 60                RTS               ;OTHERWISE, RETURN
03AA ADF317      DEHALF LDA CNTH30       ;DELAY 1/2 BIT TIME
03AD 8DF417            STA TIMH          ;BY DOING A DOUBLE
03B0 ADF217            LDA CNTL30        ;RIGHT SHIFT OF
03B3 4A                LSR               ;THE COUNT VALUES
03B4 4EF417            LSR TIMH          ;AND THEN
03B7 90E3              BCC DE2           ;COUNTING THEM DOWN
03B9 0980              ORA #$80          ;FORCE A NEGATIVE
03BB B0E0              BCS DE4           ;TO FORCE A BRANCH.
03BD 00                BRK               ;BLOCK SEPARATOR
03BE 201F03      GETBYT JSR GETCHP       ;GO GET A CHARACTER
03C1 20AC1F            JSR PACK          ;MAKE IT A NIBBLE
03C4 201F03            JSR GETCHP        ;GET ANOTHER CHARACTER
03C7 20AC1F            JSR PACK          ;STUFF IT WITH THE OTHER
03CA A5F8              LDA INL           ;GET THE WHOLE THING
03CC 60                RTS               ;AND RETURN
03CD A207        CRLFD LDX #$07          ;SET INDEX TO SEVEN,
03CF BDD51F      PRTST LDA TOP,X         ;OUTPUT CR,LF AND
03D2 20A01E            JSR OUTCH         ;NULLS
03D5 CA                DEX               ;COUNT THE CHARACTERS
03D6 10F7              BPL PRTST         ;LOOP UNTIL DONE
03D8 60                RTS               ;AND RETURN
03D9 00                BRK
```

assignments and a program based on the KIM routines, we can minimize the effort needed to build and program a new serial port. The program in listing 1 is basically a copy of the KIM software UART. Note that your choice of input pin will allow you to use these same routines to cause the input from the terminal or a keyboard to generate an interrupt if you so choose. This may be implemented following instructions in the *KIM User Manual* (Appendix H) for using PB7 to cause an interrupt.

Any routine which calls this serial I/O program should first call INIT - (JSR INIT), the normal KIM-1 power-up initialization routine which configures the B Application Port as output on PB0. If you use the remaining five pins of Port B for other purposes, you must override the pin assignments or change the value loaded in X by the statement at 0251_{16} to accommodate the needs of your other hardware. Once the new port has been initialized, you can use any of the routines in this program in exactly the same manner as you have previously used the similar routines from the KIM-1 monitor.

MICRO