Canada $4.25
USA $3.50

Transactor

THE MAGAZINE FOR COMMODORE PROGRAMMERS   FEBRUARY 1989: VOLUME 9, ISSUE 3

- Non-destructive windows on the C128
- Trouble-free 2400 Baud serial communication on the C64
- A DOS wedge for the C64 in ROM
- Far-Sys: execute machine code in *any* RAM area
- An easy-to-build parallel printer in terface for the C128
- Special centrespread bonus: GEOS label reference chart
- Machine language routines for game programming
- Serial I/O routines in Power C
- All about the C128's BANK command
- **Product Reviews**: Z3PLUS for CP/M, JiffyDOS, SWL shortwave decoder, The ZR2 Hardware interfacing chip
- **Plus** Regular columns by Todd Heimarck and Joel Rubin, Programming tips in *Bits,* and more

Dreamtime by Wayne Schmidt

0 29074 12121 02

Transactor

The Magazine for Commodore Programmers

Volume 9, Issue 3

Publisher
Antony Jacobson

Vice-President Operations
Jeannie Lawrence

Vice-President Advertising
Maximillian Jacobson

Assistant Advertising Manager
Julie Cale

Editors
Malcolm O'Brien
Nick Sullivan
Chris Zamara

Contributing Writers
Marte Brengle
Paul Bosacki
Bill Brier
Anthony Bryant
Joseph Buckley
Jim Butterfield
William Coleman
Richard Curcio
Miklos Garamszeghy
Larry Gaynier
Todd Heimarck
Adam Herst
Robert Huehn
George Hug
Zoltan Hunt
Garry Kiziak
Francis Kostella
Mike Mohilo
D.J. Morriss
Noel Nyman
Adrian Pepper
Steve Punter
Joel Rubin
David Sanner
Stephen Shervais
Audrys Vilkas
W. Mat Waites

Cover Artist
Wayne Schmidt

Toward 2400

RS-232 revisited

by George Hug

The performance of 2400-baud modems with C64s and C128s will benefit from a new look at the RS-232 servicing routines. That performance is poor at 1 MHz, and errors occur even at 2 MHz when data flows continuously or in both directions at once. The 64 and 128-mode RS-232 drivers (which are almost identical) are inefficient and contain several outright bugs. There is even a hardware glitch in many 6526 CIA chips. New routines overcoming these faults will permit error-free communication at 2400 baud at either CPU speed.

Commodore RS-232

The RS-232 drivers send and receive data one bit at a time. At 2400 baud the transmit driver runs Timer A of CIA#2 in continuous mode with a latch value of 426 (the 1-MHz I/O clock divided by 2400). On each timeout, the NMI service routine places the next bit of outgoing data on pin M (PA2) of the User Port. Ten NMIs must be serviced to transmit one byte of 8/N/1 data.

Timer B is used for received data, which enters on User Port pins B (FLAG) and C (PB0). The high-to-low transition at the beginning of the start bit generates a FLAG NMI. In response, the service routine disables the FLAG NMI, enables the Timer B NMI, and sets Timer B to time out at the mid-point of the start bit. (The service code itself uses 100 of the 213 cycles in a half-bit, leaving a timer load value of 113.) When the NMI occurs, Timer B is set to time out every 426 cycles so that pin C can be sampled at the mid-point of each bit period. At mid-stop-bit the NMIs are switched back to start-bit detection mode - FLAG enabled, Timer B disabled. Eleven NMIs must be serviced to receive one byte of data.

RS-232 inefficiencies

The following characteristics do not produce errors as such, but needlessly limit the baud rate attainable at a given CPU speed. All relate to the receive function.

1. During each byte, 450 clock cycles are consumed in manually re-starting Timer B once per bit. The re-start routine - located at $fedd ($e87f in 128 mode) - determines how long ago the timeout occurred, adds an allowance for its own execution time, subtracts that sum from the bit time, and loads the timer with the difference. After the timer is started, its reload latch is reset to $ffff. This appears to be a software emulation of the VIC 20's 6522 VIA chip. The VIA's Timer B has no continuous mode, but its one-shot mode underflows to $ffff and continues counting down. Since the CIA's Timer B does operate in continuous mode, the VIA emulation seems to be pointless.

2. The RS-232 driver is biased toward a late sampling of pin C. The sampling point is 70 cycles late in the first place because of code execution time between the mid-bit NMI and the actual sampling. (In 128 mode, sampling begins 91 cycles late, but works back to near mid-point at a rate of 12 cycles per NMI.) In addition, since the VIA emulation manually restarts Timer B, any video DMA during that process may cause a permanent, cumulative, 40-cycle delay in all subsequent samplings of the current byte. Finally, the actual pin-C data rate of a 1200- or 2400-baud modem may range from the nominal baud rate to as much as 1.6% fast. The combination of late sampling points and short bit periods may result in a sampling past the end of a bit period.

3. Continuous data flow hits a bottleneck at the junction of the stop and start bits, where three NMIs occur within one bit period. For example, the mid-stop-bit NMI requires 287 clock cycles to service (325 cycles in 128 mode - the extra time results from saving and restoring the current bank), but at 2400 baud the next byte may start after only 213 of those cycles. A related limitation is the 2224 cycles (2639 cycles in 128 mode) of total NMI service time needed to receive one byte of data. At 1 MHz, continuous 2400-baud inflow requires 55% (66%) of available clock cycles just to service NMIs.

RS-232 bugs

The defects described below cause errors without regard to baud rate, mode or CPU speed.

1. The routine at $f0a4 ($e7ec in 128 mode) disables all RS-232 activity so that NMIs will not corrupt disk, tape or REU access. It is called by the KERNAL routines LOAD, SAVE, OPEN, CHKIN, CHKOUT, LISTEN and TALK for serial bus devices and the datasette. It should be called, but is not, by DMACALL - the 128's REU routine at $ff50.

```
f0a4  pha
f0a5  lda $02a1    ;copy of enabled NMIs
f0a8  beq $f0bb    ;none enabled - done.
f0aa  lda $02a1    ;any current activity?
f0ad  and #$03     ;TA(b0) or TB(b1) enabled?
f0af  bne $f0aa    ;yes, test until idle
f0b1  lda #$10     ;no, awaiting start bit
f0b3  sta $dd0d    ;disable FLAG NMI (b4)
f0b6  lda #$00     ;all off now (?)
f0b8  sta $02a1    ;update copy
f0bb  pla
f0bc  rts
```

The $f0aa-f0af sequence pauses until the transmit buffer has been emptied and any incoming byte has been received. Then at $f0b3 it turns off the start-bit detector. However, if an incoming start-bit edge should arrive after $f0aa but before $f0b3, the resulting NMI servicing will disable FLAG (making $f0b3 redundant) and enable the Timer B NMI. Since $f0b8 clears only the mask copy, the Timer B NMI will indeed take place, with unpredictable results.

2. In the BSOUT routine the buffer pointer is incremented (at $f020/$e768) before the byte to be transmitted is placed in the buffer (at $f026/$e76e). If the NMI service routine comes looking for that new byte in the interim, it will transmit the wrong character.

3. The routine at $ef3b ($e67f) is used by the NMI service routines, and by CHKIN and BSOUT, to enable or disable an NMI source, as specified in the accumulator.

```
ef3b  sta $dd0d    ;enable or disable the NMI
ef3e  eor $02a1    ;change copy to match
ef41  ora #$80     ;enable bit on
ef43  sta $02a1    ;update copy
ef46  sta $dd0d    ;enable masked NMIs
ef49  rts
```

The routine is executed while NMIs are enabled. Should an NMI of the opposite "direction" occur after $ef3e and before $ef46, the resulting servicing may change the NMI enabled for that direction. Upon the return, however, $ef43 or $ef46, or both, will restore the old (wrong) NMI. This error occurs only when transmission takes place in both directions simultaneously.

4. It appears that many 6526 CIA chips have a hardware defect involving the interrupt flag for Timer B. If Timer B times out at about the same time as a read of the interrupt register, the Timer B flag may not be set at all. Under the VIA emulation, Timer B will then underflow and count down $ffff cycles before generating another NMI. A whole series of incoming bytes may be lost as a result. The defect was present in five of six C128s and two of three C64s sampled. When "good" and "bad" chips were switched, the problem followed the "bad" chip. There appear to be no such defects with respect to the flags for Timer A or FLAG. This glitch can cause errors during simultaneous I/O - when Timer A generates the NMI and Timer B times out just as the service routine reads $dd0d.

A software solution

The most demanding performance standard for full-duplex RS-232 is the error-free processing of continuous, bidirectional, asynchronous transmission ("CBAT"), meaning that data streams generated by unrelated clocks flow, without pause, in both directions at the same time. Fortunately, such performance at 2400 baud is attainable through software, even at 1 MHz. The approach presented here retains bit-by-bit servicing, but adopts a few key simplifications, beginning with elimination of two receive NMIs. The mid-start-bit NMI exists only to check for a false start bit, which for technical reasons would never be detected on a PSK/QAM modem. The mid-stop-bit NMI tests for a framing error, or missing stop bit, which is ignored by most software.

Another change is the removal from the NMI service routine of all matters related to parity, x-line handshake, half-duplex transmission, multiple stop bits, and the RSSTAT framing, parity, overrun and break errors. All such items take up time, are seldom used, and can be implemented separately if really needed. Finally, the VIA emulation is discarded.

New Modem Routines

Program 1 ("newmodem.src") is generic assembly language source code for a collection of new RS-232 routines. The code is not a patch to any specific BBS or terminal software, but rather one example of what might be installed by the author of such a program, or by one having access to its source code. The assembled code uses less than two pages of memory. In 128 mode it must be visible in bank 15.

The new NMI routine begins at line 3000 by pushing the registers onto the stack (already done in 128 mode, which enters at 3050). Lines 3060-3170 determine which enabled NMI sources have triggered. The 6526 glitch is finessed by comparing the high byte of Timer B before (3060) and after (3110) the read of the interrupt register (3090). If the value is higher after the read than before, then Timer B must have timed out during that period. Line 3140 makes sure B's flag bit is set in the accumulator, and 3150 makes sure it is cleared in $dd0d.

Beginning at 3180 the routine is structured to accommodate CBAT. The NMI routine does only a few critical operations while the NMIs are disabled, saving its "housekeeping" chores for later. That prevents a new NMI (one occurring after 3090) from going unserviced for too long. The critical operation for the Timer A NMI is placement of the next outgoing bit on pin M (3200-3230). The FLAG NMI must load Timer B with the start-bit timer value and start it counting down (3270-3320). The Timer B NMI must sample pin C (3120). (Pin C is sampled on every NMI; the sampling is ignored if Timer B is not an NMI source.) Once these operations have been completed, the NMIs are re-enabled (3360 or 3470).

Housekeeping chores for the Timer A NMI (3720-3920) include isolating the next output bit, or fetching the next byte from the transmit buffer, or stopping Timer A and disabling its NMI if the buffer is empty. (Timer A is loaded and started, and its NMI enabled, only by BSOUT.) In FLAG housekeeping (3330-3420), the FLAG NMI is disabled and the Timer B NMI enabled, the Timer B reload latch is loaded with the full-bit timer value, and the bit counter is initialized. Timer B housekeeping (3510-3680) processes the sampled pin-C value. If the last data bit has been received, the new byte is stored in the receive buffer, Timer B is stopped, and the NMIs are prepared for a new start bit - FLAG enabled, Timer B disabled.

The procedure at 3630 is used to change the enabled NMIs. It disables all NMIs, calculates the new configuration, and then enables that configuration. The duplicate disabling instructions at 3640/50 are necessary because an NMI occurring during the first one will be serviced immediately thereafter, resulting in re-enabled NMIs which must be disabled again by the second (there is nothing left to interrupt the second).

Following the new NMI routine are replacements for the defective routines described earlier. A new DISABL at line 4000 is a substitute for the old one at $f0a4/e7ec. Since the old one cannot be re-vectored, a call to DISABL should be made before any disk, tape or REU operation if there is any chance that the modem might generate an NMI. The NBSOUT routine at 5000 is a new front end for BSOUT which corrects the buffer pointer problem and avoids a call to $ef3b/e67f. A direct call to RSOUT (5050) will send a character to the modem regardless of the current output device.

NCHKIN at 6000 is a new front end for CHKIN which avoids $ef3b/e67f. A direct call to INABLE (6070) will re-enable the RS-232 input function without also selecting device #2 for input. Either NCHKIN (to #2) or INABLE must be called after disk, tape or REU operations to re-enable start bit detection. The BAUD section (6090-6190) sets the receive baud rate by poking the correct timer values into the NMI service code. It assumes that the current baud rate is already reflected in the BAUDOF variable at $299 ($a16), and selects one of three baud rates (2400, 1200, or 300) based on the high byte value of BAUDOF. If NCHKIN (to #2) or INABLE will be called frequently, BAUD should be moved to a separate routine which is called only after OPEN or when the baud rate needs to be changed. Provision could also be made for additional baud rates if needed.

RSGET at 7000 will fetch a character from the RS-232 input buffer regardless of the current input device. It differs from GETIN in that it does nothing to RSSTAT but instead returns with the carry flag set if the buffer was empty.

The SETUP routine at 2000 points the relevant page 3 vectors to the new NMI, NCHKIN and NBSOUT code. SETUP is the first entry in the jump table (1530). Also included in the jump table are the non-vectored routines INABLE, DISABL, RSGET and RSOUT. Finally, the receive start-bit and full-bit timer values for the three baud rates are located in a table beginning at 1590.

Calibration and performance

The new NMI routine was tested under CBAT conditions to establish the receive timer values which work for various combinations of computer, CPU speed, video DMA activity and modem speed. The tests made use of the fact that a 50%-duty-cycle square wave also constitutes continuous transmission of the letter 'u' (%01010101) in RS-232 8/N/1 format. The square wave was generated using the serial port of CIA#1, the clock output of which (CNT1) is available at pin 4 of the User Port. A spare card-edge connector (Cinch #50-24A-30) was installed in the User Port with pins 4, B and C wired together.

Program 2 ('calibrate') was used to run the tests. It keeps the CNT1 "modem" clocking continuously by feeding new output to the CIA#1 serial port during the IRQ routine. It parks in a GETIN loop which prints an '*' to the screen if a received character is not a 'u'. Program 2 also provides for continuous transmission by filling the output buffer with u's and changing line 3820 to read, in effect, 'beq getbuf'.

Timer values for the receive start bit (sb) and full bit (fb), the CNT1 "modem" (cn), and the transmit function (tx) are set in line 210 of Program 2 for each trial, which consists of running the program and looking for asterisks. If none appear then CBAT processing is error-free at those settings. One minute is enough to run through the possible overlaps of transmit and receive NMIs, and video DMAs if enabled. Asynchronous timing is approximated if the fb, cn, and tx values are different.

Table 1 shows the 2400-baud test results with the tx rate fixed at 2400 and the fb rate fixed midway between 2400 and 1.6% fast. For each hardware combination, the tests determined the highest and lowest start-bit times (sb) providing error-free CBAT. While the acceptable sb range varies with each set-up, there is a 70-cycle range, with a mid-point of 459, which works in all set-ups. Any change to the new NMI routine would require re-calibration, and the results might be different.

Table 2 compares the NMI service times required under the old and new routines. Reductions are particularly dramatic in the receive function.

Program 3 ('ciatest64') tests for the glitch in Timers A and B of CIA#2. Load and run in 64 mode only, without the card edge connector. Only a Timer B glitch has been found so far.

For transmission in only one direction at a time, the 'newmodem' routines should be replaced with shorter, faster ones. The "simultaneous" bugs will no longer occur, separate routines for each NMI type can be vectored in at $318/319 in sequence, and NMIs need not be disabled during servicing. Much higher baud rates can be attained under those conditions.

Random thoughts

1. The usual caveats apply about cartridges, special ROMs, IEEE drivers, and connecting anything homemade to the User Port.

2. CIA chips produce a count equal to the timer load value plus one. So a 425 timer value is really 1022727/426 = 2400.8 baud.

3. The SLOW command turns on the video DMAs even in 80-column mode (the 40-column screen shows a border). Turn off the DMAs by clearing the blanking bit - bit 4 of $d011. Program 2 does that through variable dm.

4. New drivers will not cure aborted Xmodem or Punter transfers caused by running 1 MHz transfer routines at 2 MHz, but they will permit the routines to be run at 1 MHz without modem errors.

5. Program 1 starts and stops the timers and also enables and disables their NMIs. If nothing else uses the timers, the NMIs could be left enabled. Time also might be saved by having the transmit NMIs occur only when the level on pin M needs to change, or at the stop bit, whichever occurs first.

Table 1: Calibration Results for 2400 Baud.

Computer mode	64	128	128	128
CPU speed (MHz)	1	1	1	2
Display mode	40	40/80	80	80
Video DMAs	on	on	off	off
TX (Tx bit time)	425	425	425	425
FB (Rx full bit)	421	421	421	421
Nominal modem:				
CN (CNT1 "modem")	426	426	426	426
Low SB (Rx start)	394	392	330	424*
High SB	568	538	618	724
Fast modem:				
CN	418	418	418	418
Low SB	350	348	290	354
High SB	524	494*	580	688

* Most restrictive. Mid-point = 459.

Table 2: NMI Service Times (cycles per byte).

	64 Mode		128 Mode	
	OLD	NEW	OLD	NEW
Transmit:				
Data bits 1-8	1320	1192	1624	1360
Stop bit	196	148	234	169
Start bit	179	173	217	194
Total	1695	1513	2075	1723
Receive:				
FLAG	157	153	195	174
Start bit	188	-	223	-
Data bits 1-7	1393	959	1659	1106
Data bit 8	199	185	237	206
Stop bit	287	-	325	-
Total	2224	1297	2639	1486

Program 1: Source code for the new serial modem routines.

```
1100 ;---------------------------------------------
1110 ;  "newmodem.src" - 64 mode.
1120 ;  @128 = changes for 128 mode.
1130 ;---------------------------------------------
1140 ribuf  =$f7          ;@128 $c8
1150 robuf  =$f9          ;@128 $ca
1160 baudof =$0299        ;@128 $0a16
1170 ridbe  =$029b        ;@128 $0a18
1180 ridbs  =$029c        ;@128 $0a19
1190 rodbs  =$029d        ;@128 $0a1a
1200 rodbe  =$029e        ;@128 $0a1b
1210 enabl  =$02a1        ;@128 $0a0f
1220 rstkey =$fe56        ;@128 $fa4b
1230 norest =$fe72        ;@128 $fa5f
1240 return =$febc        ;@128 $ff33
1250 oldout =$f1ca        ;@128 $ef79
1260 oldchk =$f21b        ;@128 $f10e
1270 findfn =$f30f        ;@128 $f202
1280 devnum =$f31f        ;@128 $f212
1290 nofile =$f701        ;@128 $f682
1500 ;---------------------------------------------
1510 *      =$ce00        ;@128 $1a00
1520 ;---------------------------------------------
1530 xx00   jmp setup
1540 xx03   jmp inable
1550 xx06   jmp disabl
1560 xx09   jmp rsget
1570 xx0c   jmp rsout
1580        nop
1590 strt24 .word $01cb   ; 459 start-bit times
1600 strt12 .word $0442   ;1090
1610 strt03 .word $1333   ;4915
1620 full24 .word $01a5   ; 421 full-bit times
1630 full12 .word $034d   ; 845
1640 full03 .word $0d52   ;3410
1650 ;---------------------------------------------
2000 setup  lda #<nmi64   ;@128 #<nmi128
2010        ldy #>nmi64   ;@128 #>nmi128
2020        sta $0318
2030        sty $0319
2040        lda #<nchkin
2050        ldy #>nchkin
2060        sta $031e
2070        sty $031f
2080        lda #<nbsout
2090        ldy #>nbsout
2100        sta $0326
2110        sty $0327
2120        rts
2130 ;---------------------------------------------
3000 nmi64  pha           ;new nmi handler
3010        txa
3020        pha
3030        tya
3040        pha
3050 nmi128 cld
3060        ldx $dd07     ;sample timer b hi byte
3070        lda #$7f      ;disable cia nmi's
3080        sta $dd0d
3090        lda $dd0d     ;read/clear flags
3100        bpl notcia    ;(restore key)
3110        cpx $dd07     ;tb timeout since 3060?
```

```
3120         ldy $dd01     ;(sample pin c)
3130         bcs mask      ;no
3140         ora #$02      ;yes, set flag in acc.
3150         ora $dd0d     ;read/clear flags again
3160 mask    and enabl     ;mask out non-enabled
3170         tax           ;these must be serviced
3180         lsr           ;timer a? (bit 0)
3190         bcc ckflag    ;no
3200         lda $dd00     ;yes, put bit on pin m
3210         and #$fb
3220         ora $b5
3230         sta $dd00
3240 ckflag  txa
3250         and #$10      ;*flag nmi? (bit 4)
3260         beq nmion     ;no
3270 strtlo  lda #$42      ;yes, start-bit to tb
3280         sta $dd06
3290 strthi  lda #$04
3300         sta $dd07
3310         lda #$11      ;start tb counting
3320         sta $dd0f
3330         lda #$12      ;*flag nmi off, tb on
3340         eor enabl     ;update mask
3350         sta enabl
3360         sta $dd0d     ;enable new config.
3370 fulllo  lda #$4d      ;change reload latch
3380         sta $dd06     ;  to full-bit time
3390 fullhi  lda #$03
3400         sta $dd07
3410         lda #$08      ;# of bits to receive
3420         sta $a8
3430         bne chktxd    ;branch always
3440 notcia  ldy #$00
3450         jmp rstkey    ;or jmp norest
3460 nmion   lda enabl     ;re-enable nmi's
3470         sta $dd0d
3480         txa
3490         and #$02      ;timer b? (bit 1)
3500         beq chktxd    ;no
3510         tya           ;yes, sample from 3120
3520         lsr
3530         ror $aa       ;rs232 is lsb first
3540         dec $a8       ;byte finished?
3550         bne txd       ;no
3560         ldy ridbe     ;yes, byte to buffer
3570         lda $aa
3580         sta (ribuf),y ;(no overrun test)
3590         inc ridbe
3600         lda #$00      ;stop timer b
3610         sta $dd0f
3620         lda #$12      ;tb nmi off, *flag on
3630 switch  ldy #$7f      ;disable nmi's
3640         sty $dd0d     ;twice
3650         sty $dd0d
3660         eor enabl     ;update mask
3670         sta enabl
3680         sta $dd0d     ;enable new config.
3690 txd     txa
3700         lsr           ;timer a?
3710 chktxd  bcc exit      ;no
3720         dec $b4       ;yes, byte finished?
3730         bmi char      ;yes
3740         lda #$04      ;no, prep next bit
3750         ror $b6       ;(fill with stop bits)
3760         bcs store
3770 low     lda #$00
3780 store   sta $b5
3790 exit    jmp return    ;restore regs, rti
3800 char    ldy rodbs
3810         cpy rodbe     ;buffer empty?
3820         beq txoff     ;yes
3830 getbuf  lda (robuf),y ;no, prep next byte
3840         inc rodbs
3850         sta $b6
3860         lda #$09      ;# bits to send
3870         sta $b4
3880         bne low       ;always - do start bit
3890 txoff   ldx #$00      ;stop timer a
3900         stx $dd0e
3910         lda #$01      ;disable ta nmi
3920         bne switch    ;always
3930 ;---------------------------------------------
4000 disabl  pha           ;turns off modem port
4010 test    lda enabl
4020         and #$03      ;any current activity?
4030         bne test      ;yes, test again
4040         lda #$10      ;no, disable *flag nmi
4050         sta $dd0d
4060         lda #$02
4070         and enabl     ;currently receiving?
4080         bne test      ;yes, start over
4090         sta enabl     ;all off, update mask
4100         pla
4110         rts
4120 ;---------------------------------------------
5000 nbsout  pha           ;new bsout
5010         lda $9a
5020         cmp #$02
5030         bne notmod
5040         pla
5050 rsout   sta $9e       ;output to modem
5060         sty $97
5070 point   ldy rodbe
5080         sta (robuf),y ;not official till 5120
5090         iny
5100         cpy rodbs     ;buffer full?
5110         beq fulbuf    ;yes
5120         sty rodbe     ;no, bump pointer
5130 strtup  lda enabl
5140         and #$01      ;transmitting now?
5150         bne ret3      ;yes
5160         sta $b5       ;no, prep start bit,
5170         lda #$09
5180         sta $b4       ;  # bits to send,
5190         ldy rodbs
5200         lda (robuf),y
5210         sta $b6       ;  and next byte
5220         inc rodbs
5230         lda baudof    ;full tx bit time to ta
5240         sta $dd04
5250         lda baudof+1
5260         sta $dd05
5270         lda #$11      ;start timer a
5280         sta $dd0e
```

```
5290         lda #$81       ;enable ta nmi
5300  change sta $dd0d      ;nmi clears flag if set
5310         php            ;save irq status
5320         sei            ;disable irq's
5330         ldy #$7f       ;disable nmi's
5340         sty $dd0d      ;twice
5350         sty $dd0d
5360         ora enabl      ;update mask
5370         sta enabl
5380         sta $dd0d      ;enable new config.
5390         plp            ;restore irq status
5400  ret3   clc
5410         ldy $97
5420         lda $9e
5430         rts
5440  fulbuf jsr strtup
5450         jmp point
5460  notmod pla            ;back to old bsout
5470         jmp oldout
5480  ;---------------------------------------------
6000  nchkin jsr findfn     ;new chkin
6010         bne nosuch
6020         jsr devnum
6030         lda $ba
6040         cmp #$02
6050         bne back
6060         sta $99
6070  inable sta $9e        ;enable rs232 input
6080         sty $97
6090  baud   lda baudof+1   ;set receive to same
6100         and #$06       ;  baud rate as xmit
6110         tay
6120         lda strt24,y
6130         sta strtlo+1   ;overwrite value @ 3270
6140         lda strt24+1,y
6150         sta strthi+1
6160         lda full24,y
6170         sta fulllo+1
6180         lda full24+1,y
6190         sta fullhi+1
6200         lda enabl
6210         and #$12       ;*flag or tb on?
6220         bne ret1       ;yes
6230         sta $dd0f      ;no, stop tb
6240         lda #$90       ;turn on flag nmi
6250         jmp change
6260  nosuch jmp nofile
6270  back   lda $ba
6280         jmp oldchk
6290  ;---------------------------------------------
7000  rsget  sta $9e        ;input from modem
7010         sty $97
7020         ldy ridbs
7030         cpy ridbe      ;buffer empty?
7040         beq ret2       ;yes
7050         lda (ribuf),y  ;no, fetch character
7060         sta $9e
7070         inc ridbs
7080  ret1   clc            ;cc = char in acc.
7090  ret2   ldy $97
7100         lda $9e
7110  last   rts            ;cs = buffer was empty
```

Program 2: Calibration program for the 64 or 128.

```
PD  100 rem  "calibrate"  for 64 or 128.
MA  110 rem  connect user port pins 4, b & c.
CE  120 rem  load "newmodem" object code at p1.
DK  130 rem  for 128 mode, un-rem 230-250.
LJ  140 rem  adjust values in 210. run. * = error.
LI  150 rem  run/stop restore to end trial.
MG  160 rem  s = (1,2) mhz; dm = dma off(0), on(1).
IM  170 rem
CL  180 close 2: open 2,2,0,chr$(6)+chr$(0): ml=12288
LP  190 for i=ml to ml+116: read a: poke i,a: z=z+a: next
DO  200 if z<>15157 then print"data error": close2: end
PC  210 sb=459: fb=421: cn=418: tx=425: s=1: dm=1
EJ  220 ri=65212: bf=peek(250)*256: bo=665: p1=52736
NH  230 rem ri=65331: bf=3328: bo=2582: p1=6656
HI  240 rem slow: if s=2 then fast: goto 260
NO  250 rem if dm=0 and peek(215)then poke ml+107,234
FG  260 for i=bf to bf+255: poke i,85: next: sys p1
KL  270 a=p1+16+(tx/256 and 6): b=sb: gosub 310
IO  280 a=a+6: b=fb: gosub 310: a=bo: b=tx: gosub310
DI  290 a=251: b=cn: gosub 310: a=598: b=ri: gosub310
NP  300 poke p1+241,0: print#2,"u";: sys ml
GG  310 q=int(b/256): poke a+1,q: poke a,b-q*256: return
HI  320 data 162,   2,  32, 198, 255,  32,  39,  48
PO  330 data  32, 228, 255, 201,  85, 240, 249,  32
DI  340 data 183, 255, 208, 244, 169,  42,  32, 210
CJ  350 data 255,  76,   8,  48, 169, 255, 141,  12
EJ  360 data 220, 173,  13, 220, 108,  86,   2, 120
FA  370 data 166, 251, 164, 252, 169,   0, 141,  26
KM  380 data 208, 141,  15, 220, 169, 127, 141,  13
IA  390 data 220, 141,  25, 208, 142,   4, 220, 140
AI  400 data   5, 220, 169,  81, 141,  14, 220, 160
BL  410 data 255, 140,  12, 220, 162,   5, 173,  13
IA  420 data 220,  41,   1, 240, 249, 202, 208, 246
KJ  430 data 140,  12, 220, 169,  28, 141,  20,   3
PP  440 data 169,  48, 141,  21,   3, 169, 136, 141
IK  450 data  13, 220,  88,  96, 173,  17, 208,  41
AG  460 data 239, 141,  17, 208,  96
```

Program 3: CIA chip test for the 64.

```
LO  500 rem  "ciatest64"  for 64 mode only.
MA  510 rem  * = interrupt flag error.
HG  520 rem  reset after test.
AD  530 rem
BL  540 n=12800: for i=n to n+103: read a: poke i,a: z=z+a
OA  550 next: if z<>11949 then print"data error":end
EG  560 sys 65412: x=not x: poke 251,x and 255
OO  570 print chr$(147);"any key switches timer."
ID  580 print"testing timer ";chr$(65-x): sys n
JB  590 wait 198,7: poke 198,0: goto 560
EI  610 data 170, 169,  98, 160,   3, 141,   4, 221
JG  620 data 140,   5, 221, 142,   6, 221, 140,   7
GO  630 data 221, 169,  17, 141,  14, 221, 141,  15
EL  640 data 221, 162,   2, 160,   7,  36, 251,  48
GG  650 data   3, 202, 160,   5, 134, 252, 140,  77
OL  660 data  50, 140,  85,  50, 138,  73, 131, 162
JL  670 data  72, 160,  50, 142,  24,   3, 140,  25
EH  680 data   3, 174,  13, 221, 141,  13, 221,  96
FL  690 data  72, 138,  72, 152, 172,   7, 221,  72
FF  700 data 173,  13, 221, 216, 204,   7, 221, 176
NM  710 data  12,  13,  13, 221,  37, 252, 208,   5
JE  720 data 169,  42,  32, 210, 255,  76, 188, 254
```

PL 600 DATA 169, 84, 162, 98, 36, 251, 48,   3

Program 4: Generator for the C64 new modem routines.

```
BC  100 rem  generator for "newmod64.obj"
FL  110 n$="newmod64.obj": rem  name of program
GF  120 nd=494: sa=52736: ch=58580

(for lines 130-260, see the standard generator on page 5)

DE  1000 data  76,  28, 206,  76, 156, 207,  76,   8
IJ  1010 data 207,  76, 213, 207,  76,  41, 207, 234
IM  1020 data 203,   1,  66,   4,  51,  19, 165,   1
CI  1030 data  77,   3,  82,  13, 169,  59, 160, 206
JB  1040 data 141,  24,   3, 140,  25,   3, 169, 140
NB  1050 data 160, 207, 141,  30,   3, 140,  31,   3
NG  1060 data 169,  33, 160, 207, 141,  38,   3, 140
JA  1070 data  39,   3,  96,  72, 138,  72, 152,  72
EK  1080 data 216, 174,   7, 221, 169, 127, 141,  13
CE  1090 data 221, 173,  13, 221,  16,  77, 236,   7
JF  1100 data 221, 172,   1, 221, 176,   5,   9,   2
OF  1110 data  13,  13, 221,  45, 161,   2, 170,  74
CI  1120 data 144,  10, 173,   0, 221,  41, 251,   5
CG  1130 data 181, 141,   0, 221, 138,  41,  16, 240
JN  1140 data  47, 169,  66, 141,   6, 221, 169,   4
HH  1150 data 141,   7, 221, 169,  17, 141,  15, 221
MM  1160 data 169,  18,  77, 161,   2, 141, 161,   2
JM  1170 data 141,  13, 221, 169,  77, 141,   6, 221
NL  1180 data 169,   3, 141,   7, 221, 169,   8, 133
IN  1190 data 168, 208,  60, 160,   0,  76,  86, 254
NK  1200 data 173, 161,   2, 141,  13, 221, 138,  41
MM  1210 data   2, 240,  44, 152,  74, 102, 170, 198
HG  1220 data 168, 208,  34, 172, 155,   2, 165, 170
JP  1230 data 145, 247, 238, 155,   2, 169,   0, 141
AG  1240 data  15, 221, 169,  18, 160, 127, 140,  13
AP  1250 data 221, 140,  13, 221,  77, 161,   2, 141
NO  1260 data 161,   2, 141,  13, 221, 138,  74, 144
LB  1270 data  14, 198, 180,  48,  13, 169,   4, 102
JC  1280 data 182, 176,   2, 169,   0, 133, 181,  76
BE  1290 data 188, 254, 172, 157,   2, 204, 158,   2
JA  1300 data 240,  13, 177, 249, 238, 157,   2, 133
EE  1310 data 182, 169,   9, 133, 180, 208, 228, 162
EB  1320 data   0, 142,  14, 221, 169,   1, 208, 188
NH  1330 data  72, 173, 161,   2,  41,   3, 208, 249
JD  1340 data 169,  16, 141,  13, 221, 169,   2,  45
KC  1350 data 161,   2, 208, 237, 141, 161,   2, 104
EF  1360 data  96,  72, 165, 154, 201,   2, 208,  96
PH  1370 data 104, 133, 158, 132, 151, 172, 158,   2
GM  1380 data 145, 249, 200, 204, 157,   2, 240,  74
MN  1390 data 140, 158,   2, 173, 161,   2,  41,   1
DE  1400 data 208,  58, 133, 181, 169,   9, 133, 180
NK  1410 data 172, 157,   2, 177, 249, 133, 182, 238
FI  1420 data 157,   2, 173, 153,   2, 141,   4, 221
KI  1430 data 173, 154,   2, 141,   5, 221, 169,  17
JO  1440 data 141,  14, 221, 169, 129, 141,  13, 221
AN  1450 data   8, 120, 160, 127, 140,  13, 221, 140
KL  1460 data  13, 221,  13, 161,   2, 141, 161,   2
EB  1470 data 141,  13, 221,  40,  24, 164, 151, 165
LA  1480 data 158,  96,  32,  59, 207,  76,  45, 207
BF  1490 data 104,  76, 202, 241,  32,  15, 243, 208
FJ  1500 data  60,  32,  31, 243, 165, 186, 201,   2
NA  1510 data 208,  54, 133, 153, 133, 158, 132, 151
FG  1520 data 173, 154,   2,  41,   6, 168, 185,  16
MO  1530 data 206, 141, 114, 206, 185,  17, 206, 141
MP  1540 data 119, 206, 185,  22, 206, 141, 140, 206
FE  1550 data 185,  23, 206, 141, 145, 206, 173, 161
JB  1560 data   2,  41,  18, 208,  35, 141,  15, 221
AI  1570 data 169, 144,  76, 101, 207,  76,   1, 247
NG  1580 data 165, 186,  76,  27, 242, 133, 158, 132
IE  1590 data 151, 172, 156,   2, 204, 155,   2, 240
DG  1600 data   8, 177, 247, 133, 158, 238, 156,   2
IP  1610 data  24, 164, 151, 165, 158,  96
```

Program 5: Generator for the C128 new modem routines.

```
MN  100 rem  generator for "newmod128.obj"
NC  110 n$="newmod128.obj": rem  name of program
DG  120 nd=494: sa=6656: ch=51020

(for lines 130-260, see the standard generator on page 5)

KG  1000 data  76,  28,  26,  76, 156,  27,  76,   8
KD  1010 data  27,  76, 213,  27,  76,  41,  27, 234
IM  1020 data 203,   1,  66,   4,  51,  19, 165,   1
PN  1030 data  77,   3,  82,  13, 169,  64, 160,  26
JB  1040 data 141,  24,   3, 140,  25,   3, 169, 140
KK  1050 data 160,  27, 141,  30,   3, 140,  31,   3
JB  1060 data 169,  33, 160,  27, 141,  38,   3, 140
JA  1070 data  39,   3,  96,  72, 138,  72, 152,  72
EK  1080 data 216, 174,   7, 221, 169, 127, 141,  13
CE  1090 data 221, 173,  13, 221,  16,  77, 236,   7
JF  1100 data 221, 172,   1, 221, 176,   5,   9,   2
KF  1110 data  13,  13, 221,  45,  15,  10, 170,  74
CI  1120 data 144,  10, 173,   0, 221,  41, 251,   5
CG  1130 data 181, 141,   0, 221, 138,  41,  16, 240
JN  1140 data  47, 169,  66, 141,   6, 221, 169,   4
HH  1150 data 141,   7, 221, 169,  17, 141,  15, 221
IM  1160 data 169,  18,  77,  15,  10, 141,  15,  10
JM  1170 data 141,  13, 221, 169,  77, 141,   6, 221
NL  1180 data 169,   3, 141,   7, 221, 169,   8, 133
BM  1190 data 168, 208,  60, 160,   0,  76,  75, 250
IK  1200 data 173,  15,  10, 141,  13, 221, 138,  41
MM  1210 data   2, 240,  44, 152,  74, 102, 170, 198
DE  1220 data 168, 208,  34, 172,  24,  10, 165, 170
MN  1230 data 145, 200, 238,  24,  10, 169,   0, 141
AG  1240 data  15, 221, 169,  18, 160, 127, 140,  13
BP  1250 data 221, 140,  13, 221,  77,  15,  10, 141
IO  1260 data  15,  10, 141,  13, 221, 138,  74, 144
LB  1270 data  14, 198, 180,  48,  13, 169,   4, 102
JC  1280 data 182, 176,   2, 169,   0, 133, 181,  76
GP  1290 data  51, 255, 172,  26,  10, 204,  27,  10
LL  1300 data 240,  13, 177, 202, 238,  26,  10, 133
EE  1310 data 182, 169,   9, 133, 180, 208, 228, 162
EB  1320 data   0, 142,  14, 221, 169,   1, 208, 188
HG  1330 data  72, 173,  15,  10,  41,   3, 208, 249
JD  1340 data 169,  16, 141,  13, 221, 169,   2,  45
GC  1350 data  15,  10, 208, 237, 141,  15,  10, 104
EF  1360 data  96,  72, 165, 154, 201,   2, 208,  96
CI  1370 data 104, 133, 158, 132, 151, 172,  27,  10
PK  1380 data 145, 202, 200, 204,  26,  10, 240,  74
AO  1390 data 140,  27,  10, 173,  15,  10,  41,   1
DE  1400 data 208,  58, 133, 181, 169,   9, 133, 180
AJ  1410 data 172,  26,  10, 177, 202, 133, 182, 238
NH  1420 data  26,  10, 173,  22,  10, 141,   4, 221
BI  1430 data 173,  23,  10, 141,   5, 221, 169,  17
JO  1440 data 141,  14, 221, 169, 129, 141,  13, 221
AN  1450 data   8, 120, 160, 127, 140,  13, 221, 140
GL  1460 data  13, 221,  13,  15,  10, 141,  15,  10
EB  1470 data 141,  13, 221,  40,  24, 164, 151, 165
CK  1480 data 158,  96,  32,  59,  27,  76,  45,  27
JA  1490 data 104,  76, 121, 239,  32,   2, 242, 208
PJ  1500 data  60,  32,  18, 242, 165, 186, 201,   2
NA  1510 data 208,  54, 133, 153, 133, 158, 132, 151
MF  1520 data 173,  23,  10,  41,   6, 168, 185,  16
MC  1530 data  26, 141, 114,  26, 185,  17,  26, 141
KB  1540 data 119,  26, 185,  22,  26, 141, 140,  26
MB  1550 data 185,  23,  26, 141, 145,  26, 173,  15
PO  1560 data  10,  41,  18, 208,  35, 141,  15, 221
BM  1570 data 169, 144,  76, 101,  27,  76, 130, 246
PF  1580 data 165, 186,  76,  14, 241, 133, 158, 132
LD  1590 data 151, 172,  25,  10, 204,  24,  10, 240
LD  1600 data   8, 177, 200, 133, 158, 238,  25,  10
IP  1610 data  24, 164, 151, 165, 158,  96
```