MOTOROLA
Semiconductor Products Inc.

AN899
Application Note

A TERMINAL INTERFACE, PRINTER INTERFACE, AND BACKGROUND PRINTING FOR AN MC68000-BASED SYSTEM USING THE MC68681 DUART

Prepared by
Kyle Harper
Advanced Microcomputer Applications Engineering
Microprocessor Division
Motorola Inc.
Austin, Texas

INTRODUCTION

Very efficient terminal and printer I/O can be achieved in an MC68000-based system using only the MC68681 dual universal receiver transmitter (DUART) and an RS-232 interface driver chip set. As an extra bonus, a dual-tasking scheme can be easily implemented using the counter/timer on-chip the MC68681 to generate periodic time-slice interrupts to the MC68000. This allows the MC68000 to appear to be executing two tasks simultaneously. Typically, one of the tasks would be a printing task so that printing can be done as a "background" task to something else being executed by the MC68000.

In this Application Note, a complete MC68000/MC68681 interface and a dual-task sample application is presented. It begins with a description of the MC68681 operation and programming for this application. This is followed by a description of the MC68000/MC68681 hardware interface. Finally, the software required for the application is presented. It includes the routines required to initialize and drive the MC68681 serial channels and counter, and the software required to implement the dual-tasking scheme. The software also includes two sample task routines. One continually monitors a terminal (attached to DUART channel A) for incoming characters, assembles them into a character string in an input buffer, than places the string in a print queue. The other task continually monitors the print queue for character strings destined to be printed and sends them to the printer (attached to DUART channel B).

MC68681 OPERATION AND PROGRAMMING

The MC68681 DUART is a communications device that provides two independent full-duplex asynchronous receiver/transmitter channels, a 6-bit parallel input port, an 8-bit parallel output port, and a 16-bit counter/timer in a single package. Also, the MC68681 can be programmed to generate interrupts upon any of the following conditions:

Channel A Transmitter Ready
Channel A Receiver Ready
Channel A Change-in-Break
Channel B Transmitter Ready
Channel B Receiver Ready
Channel B Change-in-Break
Counter/Timer Ready
Input Port Change-of-State

Channels A and B of the MC68681 can operate in four different modes: normal, automatic echo, local loopback, and remote loopback. A channel operating in normal mode allows full-duplex communication. A channel operating in automatic-echo mode operates exactly as in normal mode, but automatically re-transmits any received data. Local loopback and remote loopback modes are diagnostic modes that can be used to verify correct operation of a channel.

The MC68681 has a 6-bit parallel input port and an 8-bit parallel output port. Each of the inputs and outputs can be used as general-purpose inputs and outputs. However, each has programmable alternate functions, as shown below:

Pin	Programmable Alternate Function
IP0	Channel A Clear-to-Send Input
IP1	Channel B Clear-to-Send Input
IP2	Channel B Receiver External Clock Input or Counter/Timer External Clock Input
IP3	Channel A Transmitter External Clock Input
IP4	Channel A Receiver External Clock Input
IP5	Channel B Transmitter External Clock Input
OP0	Channel A Request-to-Send Output
OP1	Channel B Request-to-Send Output
OP2	Channel A Transmitter Clock Output or Channel A Receiver Clock Output
OP3	Counter/Timer Output or Channel B Transmitter Clock Output or Channel B Receiver Clock Output
OP4	Channel A Reveiver-Ready or Buffer-Full Interrupt Output
OP5	Channel B Receiver-Ready or Buffer-Full Interrupt Output
OP6	Channel A Transmitter-Ready Interrupt Output
OP7	Channel B Transmitter-Ready Interrupt Output

Finally, the MC68681 has a 16-bit programmable counter/timer that can be used to measure elapsed time between events, or to generate periodic interrupts. It can be programmed to operate as a free-running timer (cannot be stopped and started) or as a counter (can be stopped and started).

This application will use the normal, automatic-echo, and local loopback modes, and will utilize two of the MC68681 interrupt sources: the channel A change-in-break $\overline{\text{IRQ}}$ and the counter/timer $\overline{\text{IRQ}}$. Also, one of the output port pins and one of the input port pins will be used as RTS/CTS handshake lines. In this application, a terminal will be attached to DUART channel A and will be programmed to transmit and receive at 9600 baud with seven bits/character, even parity, and two stop bits. The channel will be programmed to operate in automatic-echo mode so that the character typed at the terminal keyboard will appear on the CRT screen. So that the channel receiver FIFO is not overrun, channel A will be programmed to use the receiver RTS/CTS handshake protocol. This protocol works as follows: the receiver RTS output is connected to the CTS input of the terminal. So long as the receiver has room in its FIFO for another character, the receiver will assert RTS. If the FIFO becomes full, the receiver will negate RTS. When the FIFO once again has room for another character, it will automatically re-assert RTS. Assuming that the terminal will not transmit a character unless it sees CTS asserted, receiver overrun will not occur. Finally, the BREAK key will be used as an abort button, so that the user can exit to the monitor (or operating system) at any time. Channel A will, therefore, be programmed to generate an interrupt to the MC68000 when it receives a BREAK character from the terminal.

A printer will be attached to DUART channel B and the channel will be programmed to operate in normal mode, transmit at 300 baud with seven bits/character, even parity, and one stop bit. So that the channel does not send characters to the printer faster than the printer can handle them, channel B will be programmed to use the transmitter RTS/CTS handshake protocol. This protocol works as follows: when channel B needs to send a character to the printer, it will assert RTS and then wait for the printer to assert CTS before transmitting the character.

The MC68681 counter/timer will be programmed to generate the time-slice interrupts to the MC68000 required for dual-tasking. The counter/timer must be able to be stopped and re-started; therefore, it is programmed to operate in counter mode. After initializing the counter registers with the count value, the counter will be started. When the counter reaches terminal count, it will generate an interrupt to the MC68000. The MC68000 will then stop the counter, clear the interrupt, swap tasks being executed, and start the counter again. When the counter is started again, it will be re-initialized using the value found in the counter registers.

INTERFACE HARDWARE

The hardware required to interface the MC68681 to the MC68000 is minimal, as shown by the schematic in Figure 1. The $\overline{\text{RESET}}$, R/$\overline{\text{W}}$, and $\overline{\text{DTACK}}$ lines are connected directly between and MC68681 and the MC68000. Address lines A5-A23 are routed through address decode logic and used to generate the MC68681 chip select. Address lines A1-A4 are tied to the MC68681 register select pins RS1-RS4. The MC68681 data bus pins, D0-D7 are connected to the MC68000 lower data bus lines, D0-07. Typically, the MC68681 would be attached to the lower data bus because the MC68681 must supply an interrupt vector number to the MC68000 on D0-D7 during $\overline{\text{IACK}}$ cycles. However, if the MC68681 will not be generating interrupts, it could just as easily be attached to the upper data bus. The MC68681 $\overline{\text{IRQ}}$ line must be encoded by the SN74LS148 to give the $\overline{\text{IRQ}}$ a priority level required by the MC68000 on its IPL0-IPL2 lines. Also, the MC68000 A1-A3 lines must be decoded during $\overline{\text{IACK}}$ cycles by the SN74LS138 to generate $\overline{\text{IACK}}$ back to the MC68681. Using the SN74LS148 as the $\overline{\text{IRQ}}$ encoder and the SN74LS138 as the $\overline{\text{IACK}}$ decoder provides full support of the MC68000 seven interrupt levels. The MC68681 requires only one interrupt level. For this application, interrupt level four has been arbitrarily chosen. This leaves the other six levels for future system expansion.

The two channels are connected to the external devices via RS-232 drivers and DB-25 connectors. Because this application uses the OP0 and OP1 lines as the RTSA and CTSB handshake lines, respectively, they too are routed via the RS-232 drivers to their respective connectors.

Finally, a 3.6864 MHz crystal is connected between the MC68681 X1/CLK and X2 pins. The crystal is required for the built-in baud rate generator. The 15 pF and 5 pF shunt capacitors must also be connected between the crystal and ground as shown to insure proper operation of the baud rate generator.

INTERFACE SOFTWARE

The interface software required for this application is flowcharted in Figure 2 and is listed at the end of this Application Note. The routines can be broken down into three categories: the DUART initialization routines, the I/O driver routines, and the interrupt handling routines. The DUART initialization routines consist of DINIT, CHCHK, and CTRCHK. DINIT is the DUART initialization routine, and is called at system initialization time. After DINIT initializes the DUART channels and counter, it checks channel A, channel B, and the counter for operational errors. Before

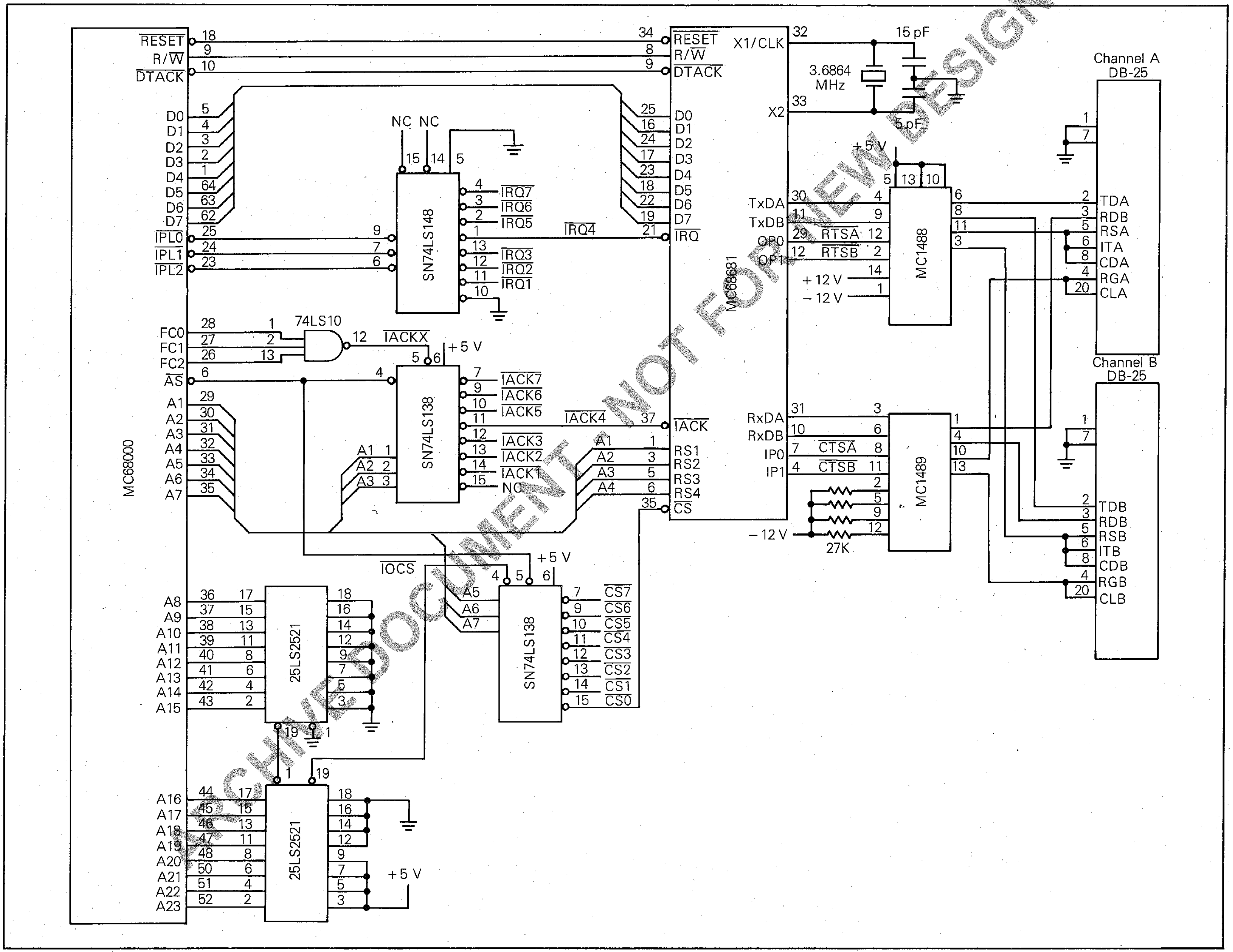


FIGURE 1 — MC68000/MC68681 Interface Schematic

DINIT is called, the calling routine must allocate three words on the system stack. Upon return to the calling routine, DINIT will pass back three status words on the system stack that reflect the operation of channel A, channel B, and the counter. If DINIT finds no errors in channel A, it will enable the channel A receiver and transmitter. Likewise, if DINIT finds no errors in channel B, it will enable the channel B transmitter. CHCHK and CTRCHK are routines that are called by DINIT to perform the actual checks. CHCHK checks a channel for proper operation. DINIT calls CHCHK twice: the first time to check channel A and the second time to check channel B. After placing the channel in local loop-back mode, CHCHK checks the channel for the following errors: transmitter never ready, receiver never ready, framing error, parity error, and incorrect character received. CTRCHK checks the counter for proper operation by verifying that the counter interrupts the MC68000 properly after reaching terminal count.

The I/O driver routines consist of INCH, OUTCH, and POUTCH. INCH is the terminal input character routine. INCH gets a character from the channel A receiver and places it in the lower byte of register D0. OUTCH is the terminal output character routine. OUTCH sends the character in the lower byte of register D0 to the channel A transmitter. POUTCH is the printer output character routine. POUTCH sends the character in the lower byte of register D0 to the channel B transmitter.

The interrupt handling routines consist of DIRQ and CIRQ. DIRQ is the DUART interrupt handling routine. After the DUART generates an interrupt, the MC68000 begins executing DIRQ. DIRQ determines whether the interrupt was caused by the counter or a channel A change-in-break. If the interrupt was caused by the counter, DIRQ causes the MC68000 to swap tasks being executed. This process is discussed in a later section. If the interrupt was caused by a channel A change-in-break interrupt (beginning of break), DIRQ clears the interrupt source, waits for the next change-in-break condition interrupt (end of break), clears the interrupt source again and then returns from exception processing to the system monitor. CIRQ is used instead of DIRQ as the DUART interrupt handling routine when CTRCHK is executing. When the counter generates an interrupt during execution of CTRCHK, CIRQ sets the carry bit in the status register, thus informing CTRCHK that the counter interrupt was generated correctly.

DUAL-TASKING SOFTWARE

The dual-tasking software required for this application is flowcharted in Figure 3 and is listed at the end of this Application Note. The routines can be broken down in two categories: the routines that facilitate dual-tasking and the two sample tasks themselves. The routines that facilitate dual-tasking consist of SWPTSKS and TSKINIT.

SWPTSKS is the task swapping routine executed when DIRQ determines that the counter generated an interrupt. SWPTSKS "swaps out" the task currently being executed with the task that is currently dormant. The "swap" process works as follows: the counter interrupt causes the MC68000 to begin exception processing. During exception processing the MC68000 stacks the active task program counter and status register on the active task system stack, then executes DIRQ. DIRQ determines that the interrupt was caused by the counter and branches to SWPTSKS. SWPTSKS stops the counter, then saves the active task register contents and user stack pointer on the active task system stack. After saving this information on the active task system stack, SWPTSK swaps out the active task system stack pointer with the dormant task system stack pointer (stored in a reserved memory location). SWPTSKS then pulls the dormant task user stack pointer and register contents off the dormant task system stack (this information was placed on the dormant system stack by a previous task swap operation), and restarts the counter. Finally, because the dormant task status register contents and program counter are now at the top of the dormant task system stack, the MC68000 will return from exception where the dormant task had been interrupted, thereby re-activating it.

TSKINIT is the task initialization routine. It initializes the DUART by calling DINIT, then checks for operational errors in the two channels and the counter. If errors are found in either of the channels or the counter, TSKINIT prints the appropriate error messages to a "command console" then stops. If no errors are found, TSKINIT then initializes the print task as the initial dormant task. The initialization procedure works like this: the dormant task system stack pointer is initialized. The start address of the print task is stacked on the system stack, then an initial status register content is stacked. This is the order in which the MC68000 requires information to be stacked when returning from exception. Next, the print task initial register contents and user stack pointer are stacked on the system stack. This is the order in which SWPTSKS requires information to be stacked to perform its task swap operation. After initializing the print task as the dormant task, TSKINIT initializes the input task user and system stack pointers, starts the counter, then begins execution of the input task.

The two sample tasks given in this Application Note are INPTTSK and PRNTTSK. The tasks work together to perform two typical I/O operations: character string input from a terminal and character string output to a printer. Because I/O hardware is character-oriented and not string-oriented, character string I/O must be transformed into character I/O by using buffers and queues. Character string input is accomplished through the use of an input buffer. Characters are placed in this buffer as they come in from the terminal. When the carriage return character is received and placed in the buffer, the string has been completely assembled and is moved elsewhere so that another one can be assembled.

Character string printing is accomplished through the use of a print buffer and a print queue. For efficient character string printing, the print buffer should be capable of holding more than one character string. This is because the MC68000 can supply strings to be printed much faster than the printer can print them. A multiple-string print buffer allows the MC68000 to "queue" character strings bound for the printer, then go on to more important things, rather than acting as a slave to the printer. The print queue is required to determine where the next string arriving at the buffer will go and where the next string departing from the buffer can be found. Print "tags" indicating that there are character strings in the print buffer are placed in this queue. The queue has an input and output pointer, and acts in a first-in-first-out manner. Thus, strings in the print buffer will be sent to the printer in the order that their print tags arrived at the print queue.

For this application, a character string is terminated by a carriage return, and maximum string length is set by the constant CSLNTH. CSLNTH is used to define the width of the input buffer and the width of the print buffer. The print queue length is set by the constant PQLNTH. PQLNTH is used to define the length of the print queue and the length of the print buffer. Both CSLNTH and PQLNTH must be assigned values that are powers of two and can have a maximum value of 256. Because maximum string length is 256 bytes, the print tags need only be a byte value.

When a character string is to be sent to the print buffer, it must be moved into the print buffer and an associated print tag placed in the print queue. When a character string is to be sent to the printer, it must be taken from the print buffer and its associated print tag removed from the print queue.

INPTTSK continually monitors the terminal attached to DUART channel A for incoming characters, assembles them into a character string in the input buffer, then queues the string in the print buffer. INPTTSK consists of two routines: ISTRG and QSTRG. ISTRG is the routine that assembles characters received from the terminal (via the INCH routine) into a character string in the input buffer. QSTRG is the routine that queues the character string in the print buffer. QSTRG first checks the status of the print queue. If the queue is full, QSTRG will wait until there is room in the queue for a print tag. If the queue is not full, QSTRG will move the character string into the print buffer and place a print tag in the print queue.

PRNTTSK continually monitors the print queue for print tags. If it finds a print tag in the queue, PRNTTSK prints the string and removes the tag from the queue. PRNTTSK consists of two routines: RSTRG and PSTRG. RSTRG is the routine that releases a character string from the print buffer, and sends it to the printer via the PSTRG routine. RSTRG checks the status of the print queue. If it is empty, RSTRG will wait until a print tag appears in the queue. If the queue is not empty, RSTRG will call routine PSTRG, then remove the print tag from the print queue. PSTRG is the routine that sends a character string to the printer character-by-character (via the POUTCH routine).

SUMMARY

The frequency at which the MC68000 swaps between tasks is directly determined by the frequency at which the DUART counter generates interrupts. This is determined by the count value placed in the upper and lower counter registers. The main concern in determining the count value is making sure that the task-swapping is transparent to the user sitting at the terminal. That is, he must not be aware that he does not have the attention of the system all the time.

The system on which this application was developed performed well with the count value set at $0073. With the counter clock source programmed to be the 3.6864 MHz crystal divided-by-sixteen, this count value causes an interrupt to occur approximately every 500 microseconds.

Also, this Application Note presents the interface required for efficient poll-driven serial I/O using the MC68681 DUART. If you wish to modify this interface to support interrupt-driven I/O, no changes in the hardware are required. Only software modifications need to be made.

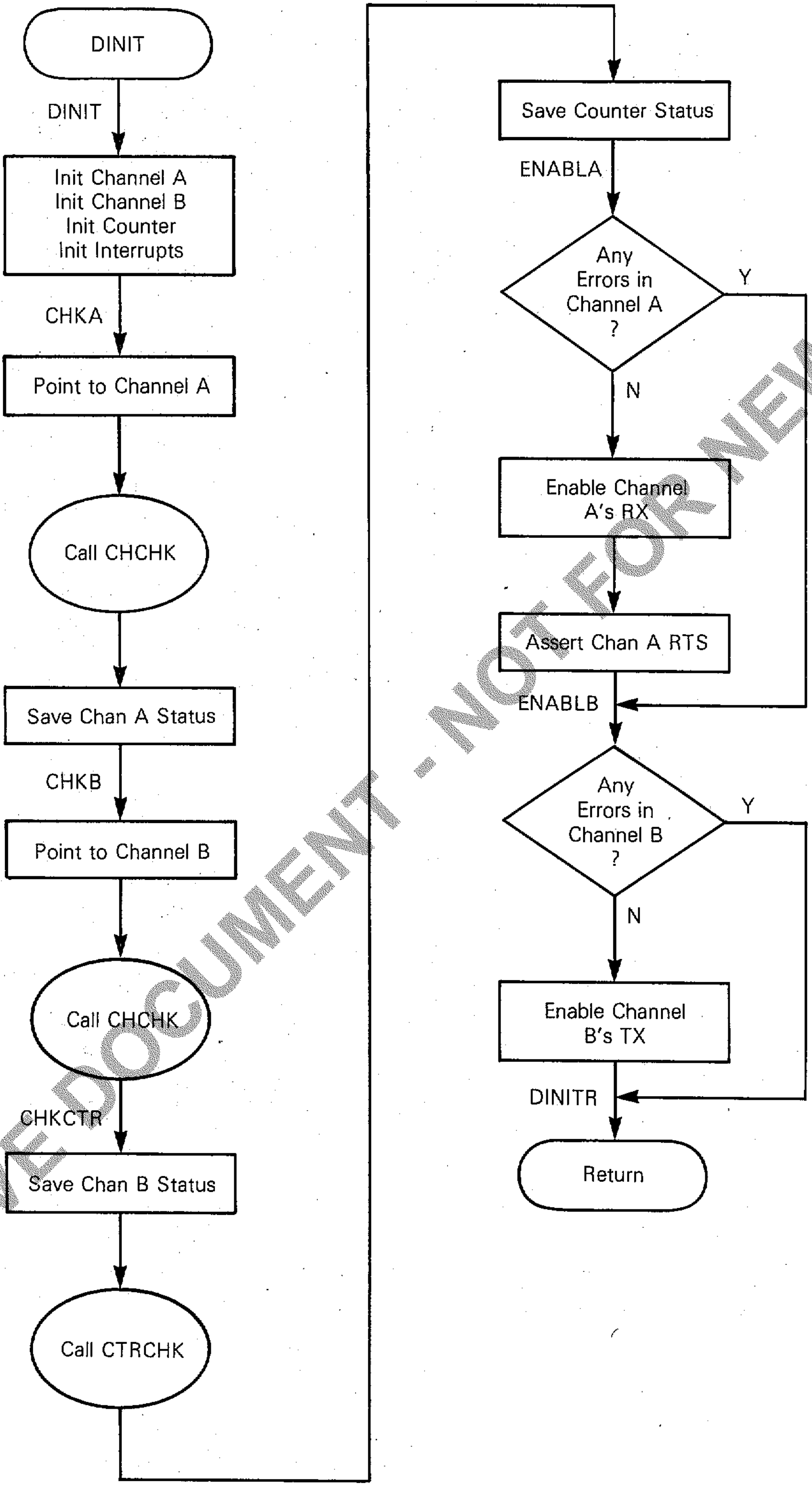


FIGURE 2 — MC68681 Interface Software Flowcharts (Sheet 1 of 6)

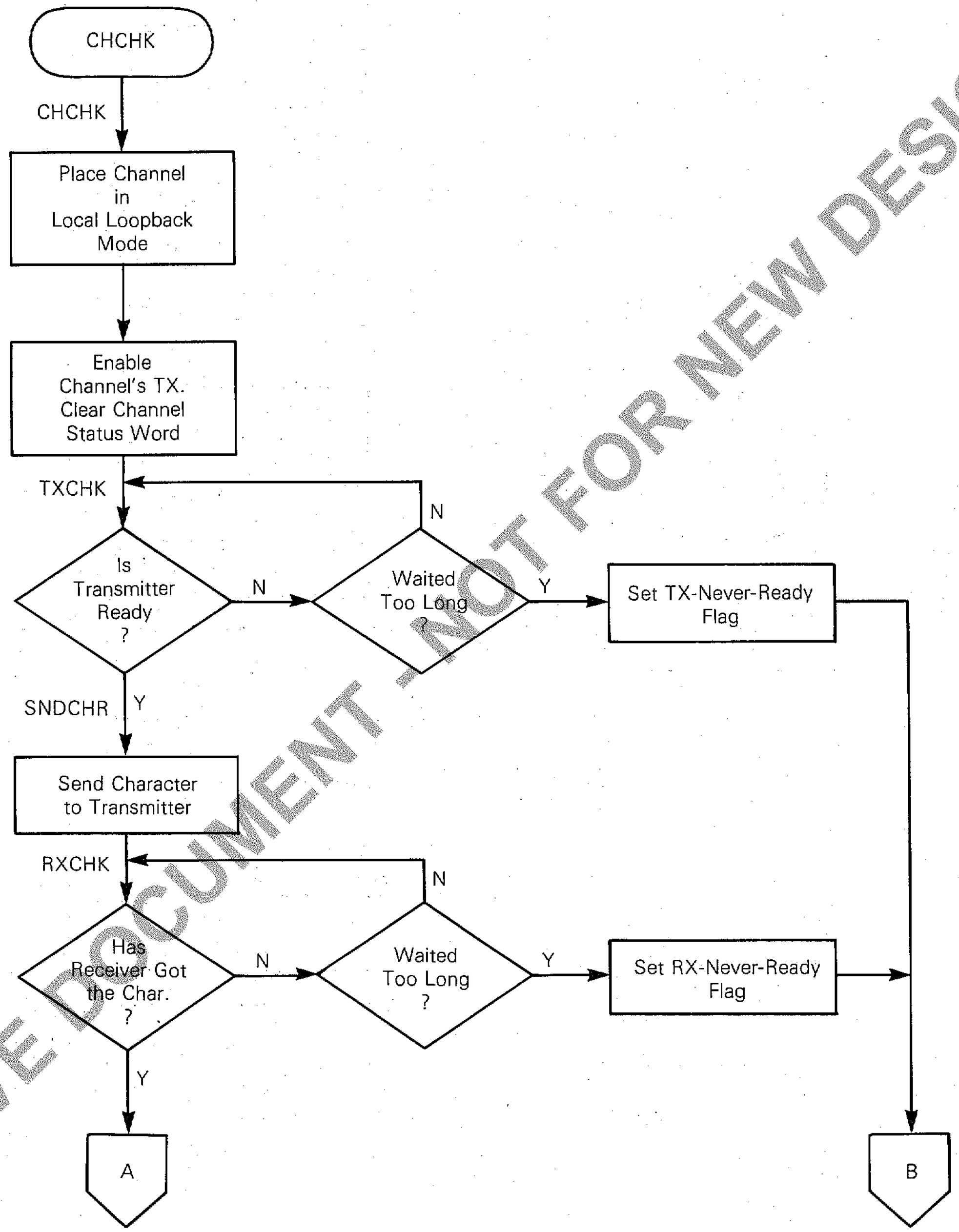

CHCHK
CHCHK
Place Channel in Local Loopback Mode
Enable Channel's TX. Clear Channel Status Word
TXCHK
Is Transmitter Ready ?
N
Waited Too Long ?
N
Y
Set TX-Never-Ready Flag
SNDCHR
Y
Send Character to Transmitter
RXCHK
Has Receiver Got the Char. ?
N
Waited Too Long ?
N
Y
Set RX-Never-Ready Flag
Y
A
B

FIGURE 2 — MC68681 Interface Software Flowcharts (Sheet 2 of 6)

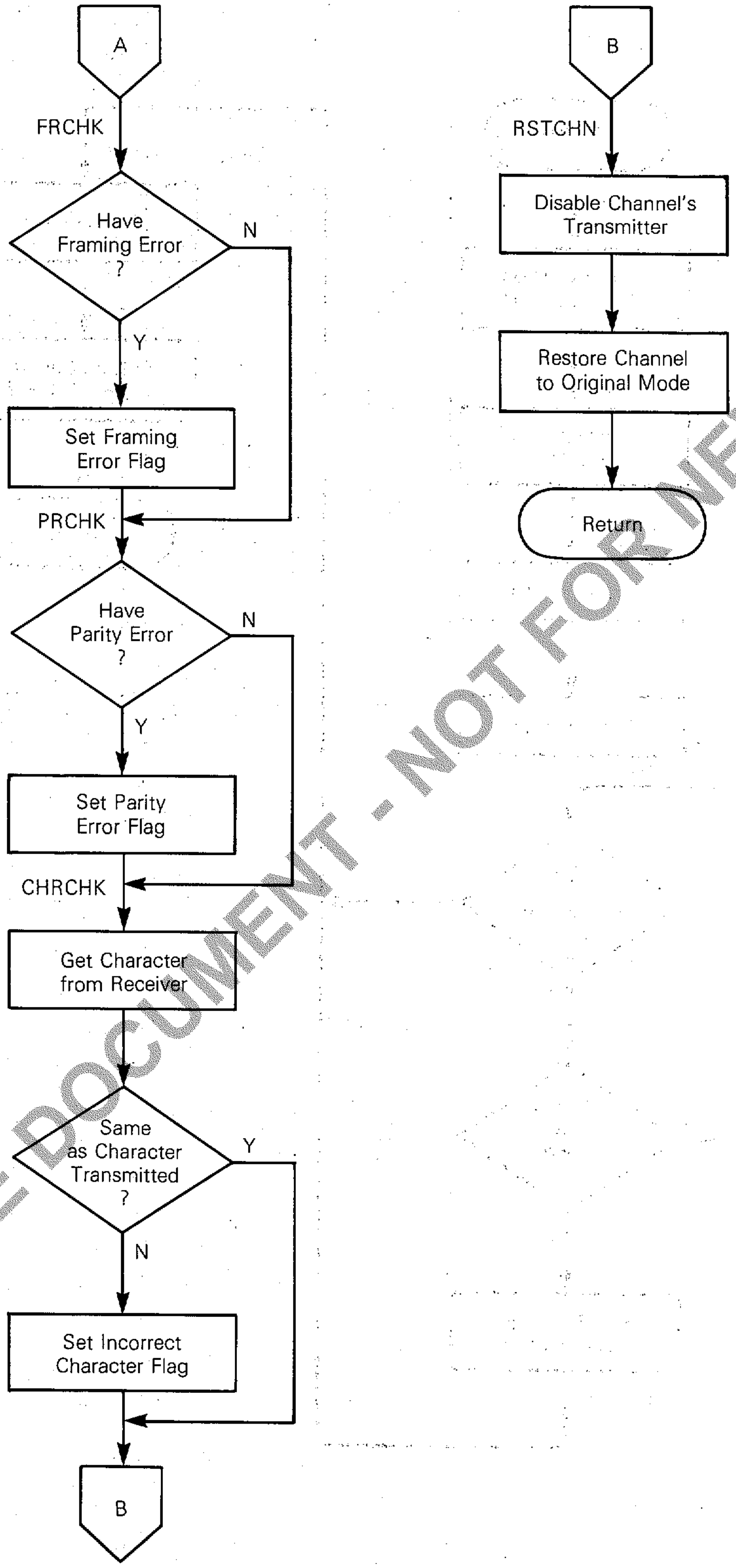

A
FRCHK
Have Framing Error ?
N
Y
Set Framing Error Flag
PRCHK
Have Parity Error ?
N
Y
Set Parity Error Flag
CHRCHK
Get Character from Receiver
Same as Character Transmitted ?
Y
N
Set Incorrect Character Flag
B
B
RSTCHN
Disable Channel's Transmitter
Restore Channel to Original Mode
Return

FIGURE 2 — MC68681 Interface Software Flowcharts (Sheet 3 of 6)

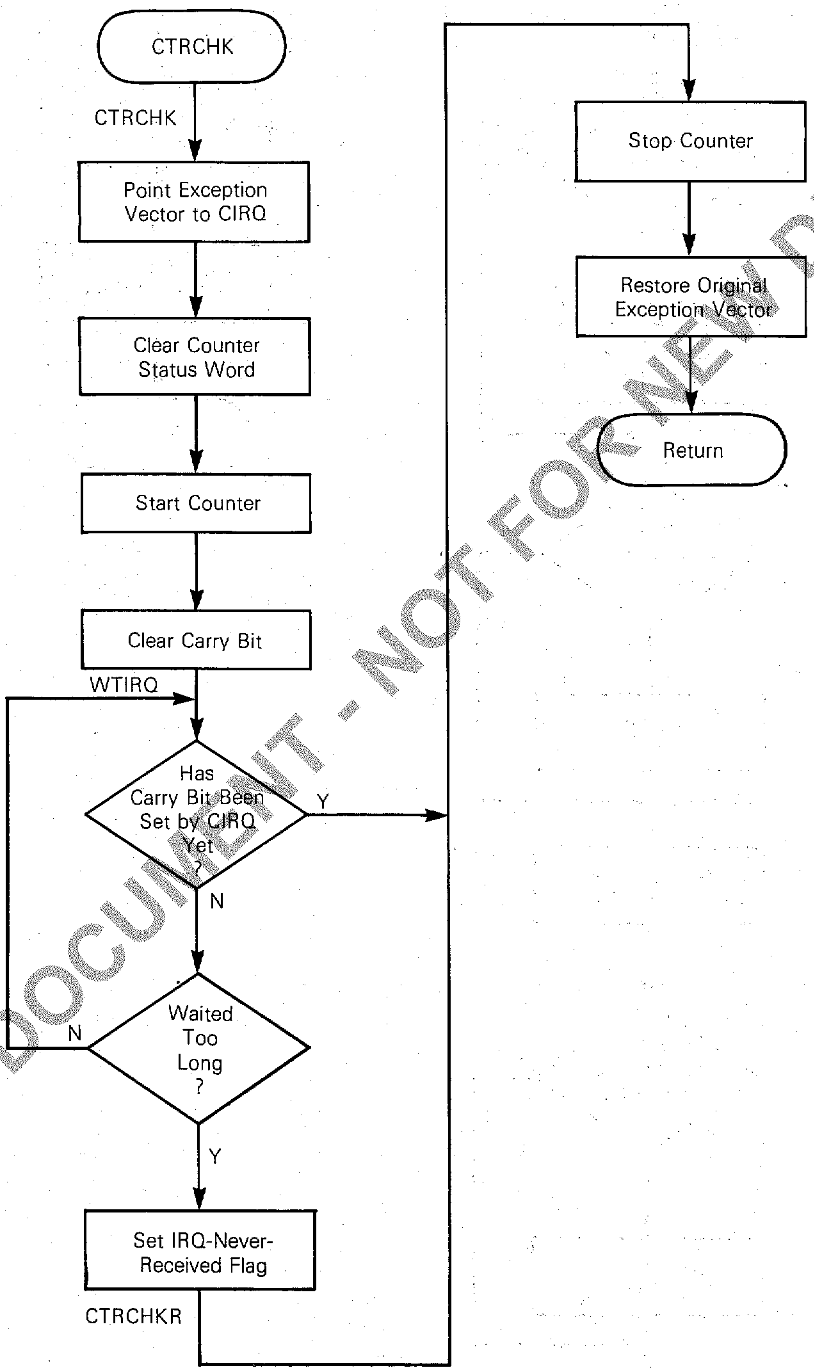

CTRCHK
CTRCHK
Point Exception Vector to CIRQ
Clear Counter Status Word
Start Counter
Clear Carry Bit
WTIRQ
Has Carry Bit Been Set by CIRQ Yet ?
Y
N
Waited Too Long ?
N
Y
Set IRQ-Never-Received Flag
CTRCHKR
Stop Counter
Restore Original Exception Vector
Return

FIGURE 2 — MC68681 Interface Software Flowcharts (Sheet 4 of 6)

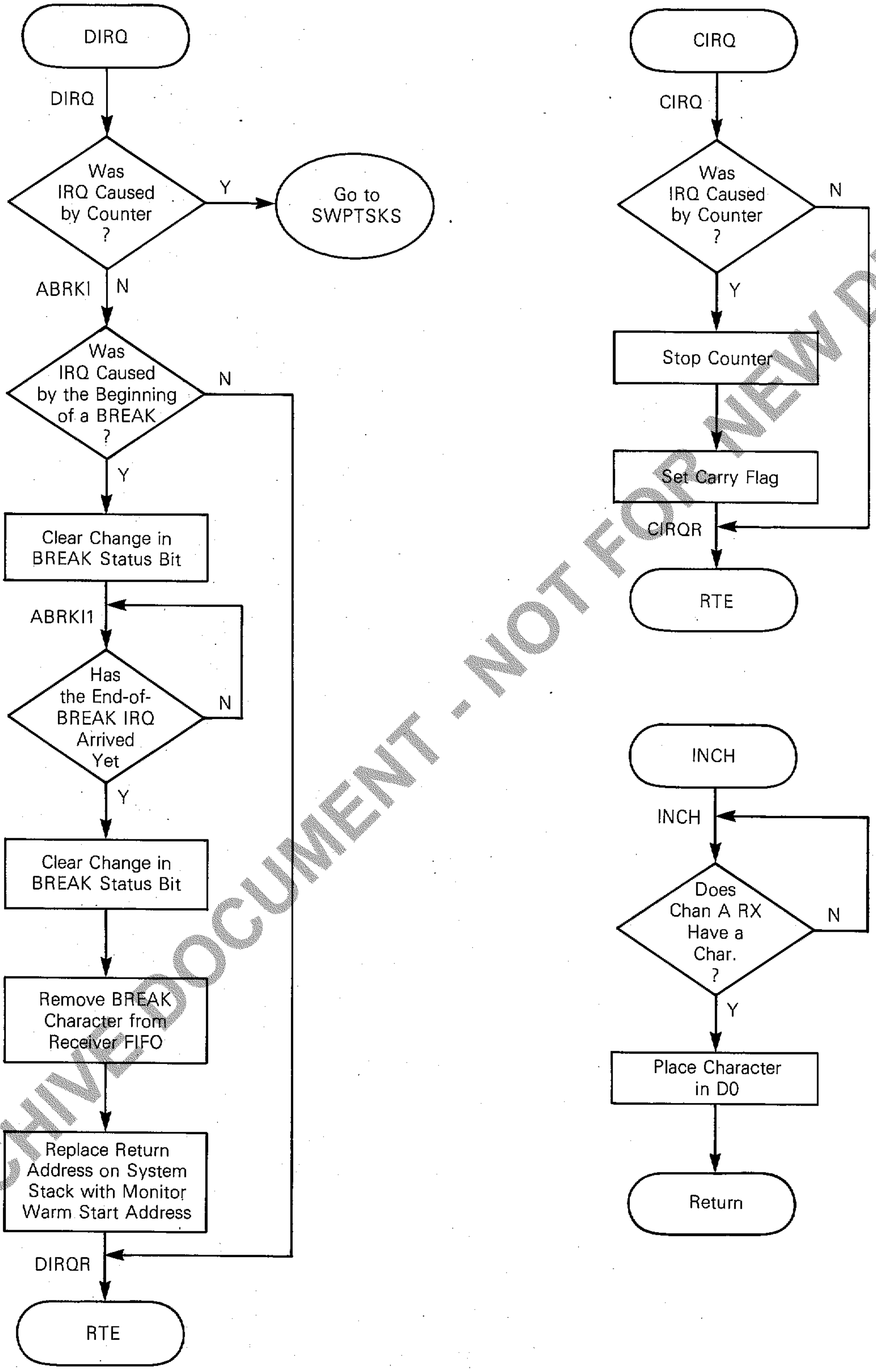
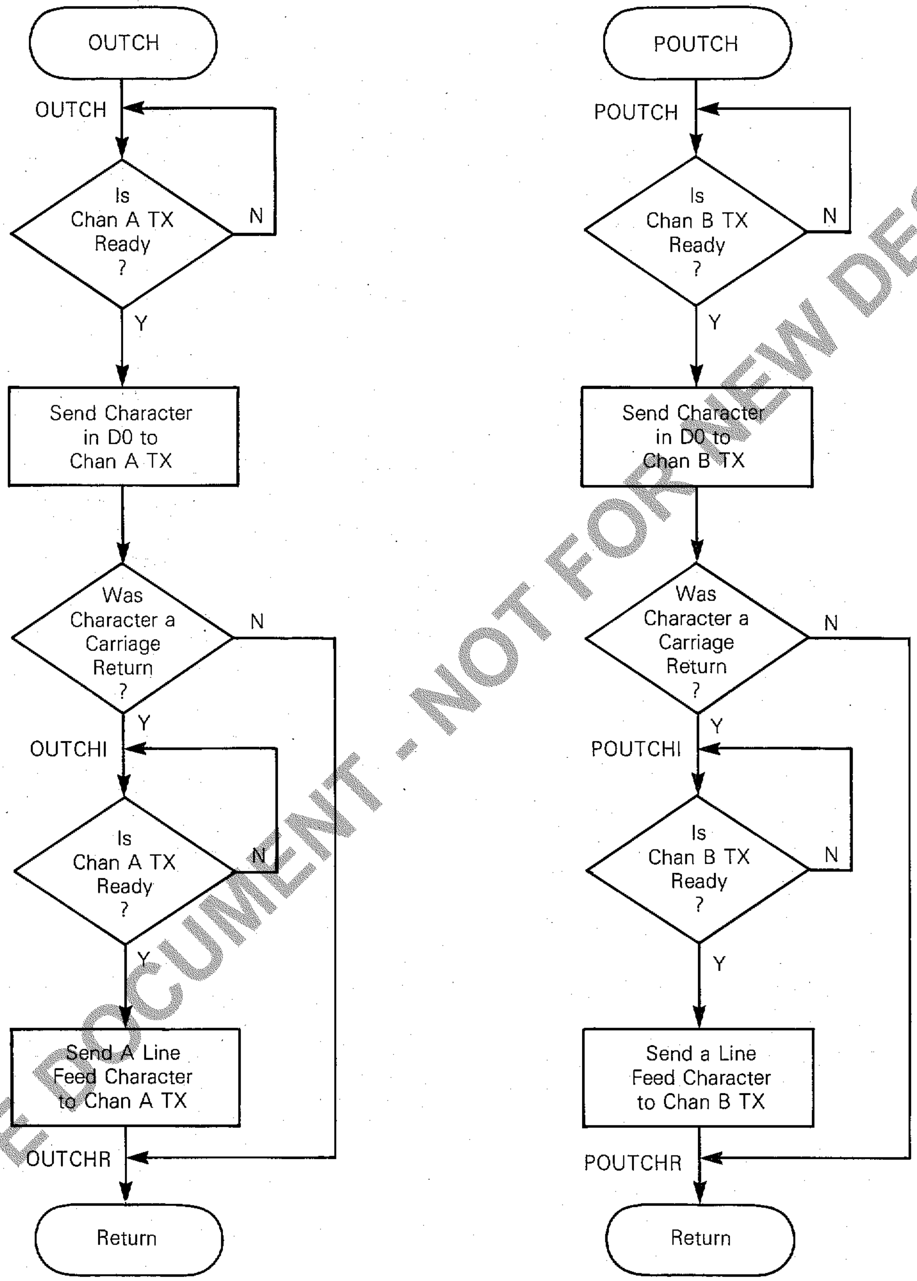

DIRQ
DIRQ
Was IRQ Caused by Counter ?
Y
Go to SWPTSKS
ABRKI
N
Was IRQ Caused by the Beginning of a BREAK ?
N
Y
Clear Change in BREAK Status Bit
ABRKI1
Has the End-of-BREAK IRQ Arrived Yet
N
Y
Clear Change in BREAK Status Bit
Remove BREAK Character from Receiver FIFO
Replace Return Address on System Stack with Monitor Warm Start Address
DIRQR
RTE
CIRQ
CIRQ
Was IRQ Caused by Counter ?
N
Y
Stop Counter
Set Carry Flag
CIRQR
RTE
INCH
INCH
Does Chan A RX Have a Char. ?
N
Y
Place Character in D0
Return

FIGURE 2 — MC68681 Interface Software Flowcharts (Sheet 5 of 6)



OUTCH
OUTCH
Is Chan A TX Ready ?
N
Y
Send Character in D0 to Chan A TX
Was Character a Carriage Return ?
N
Y
OUTCHI
Is Chan A TX Ready ?
N
Y
Send A Line Feed Character to Chan A TX
OUTCHR
Return
POUTCH
POUTCH
Is Chan B TX Ready ?
N
Y
Send Character in D0 to Chan B TX
Was Character a Carriage Return ?
N
Y
POUTCHI
Is Chan B TX Ready ?
N
Y
Send a Line Feed Character to Chan B TX
POUTCHR
Return

FIGURE 2 — MC68681 Interface Software Flowcharts (Sheet 6 of 6)

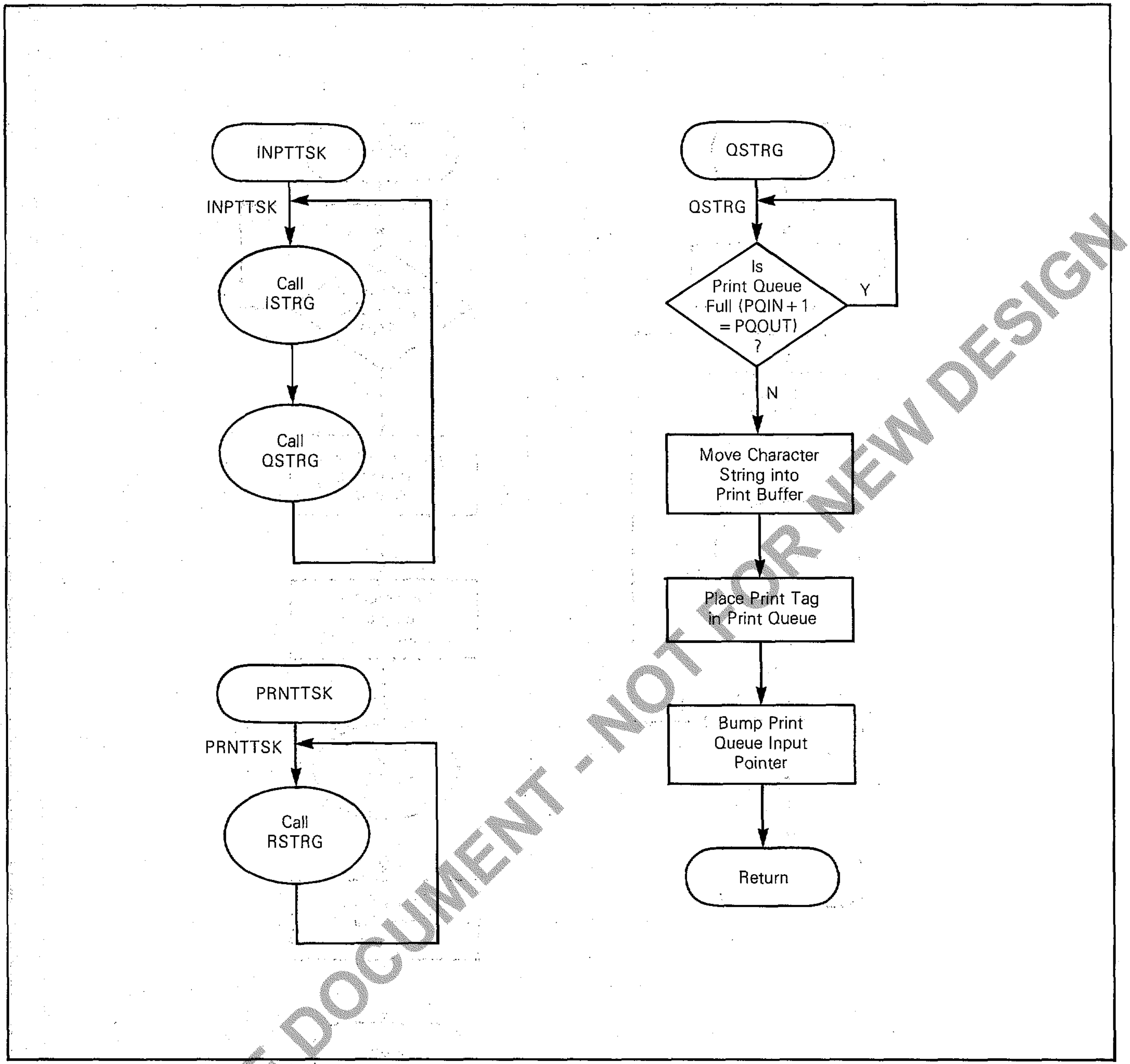

INPTTSK
INPTTSK
Call ISTRG
Call QSTRG
PRNTTSK
PRNTTSK
Call RSTRG
QSTRG
QSTRG
Is Print Queue Full (PQIN + 1 = PQOUT) ?
Y
N
Move Character String into Print Buffer
Place Print Tag in Print Queue
Bump Print Queue Input Pointer
Return

FIGURE 3 — Dual-Tasking Software Flowchart (Sheet 1 of 5)

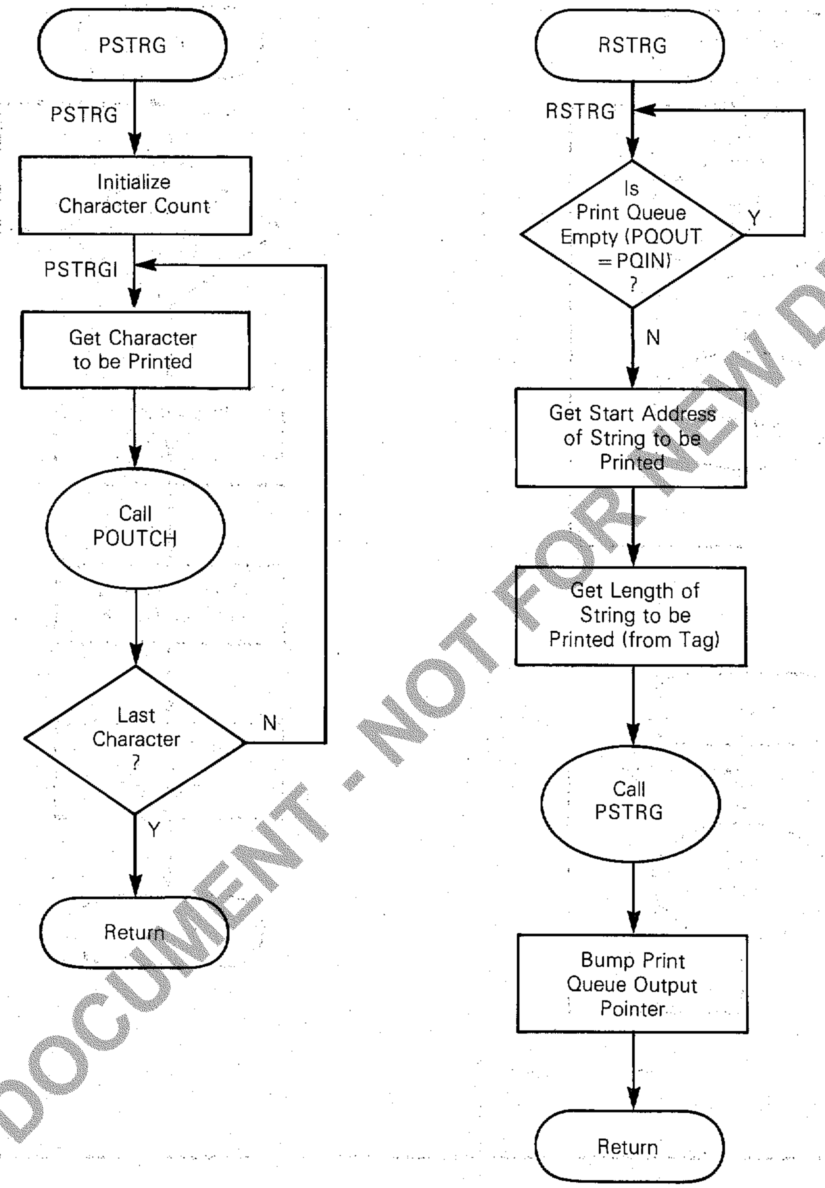

PSTRG
PSTRG
Initialize Character Count
PSTRGI
Get Character to be Printed
Call POUTCH
Last Character ?
N
Y
Return
RSTRG
RSTRG
Is Print Queue Empty (PQOUT = PQIN) ?
Y
N
Get Start Address of String to be Printed
Get Length of String to be Printed (from Tag)
Call PSTRG
Bump Print Queue Output Pointer
Return

FIGURE 3 — Dual-Tasking Software Flowcharts (Sheet 2 of 5)

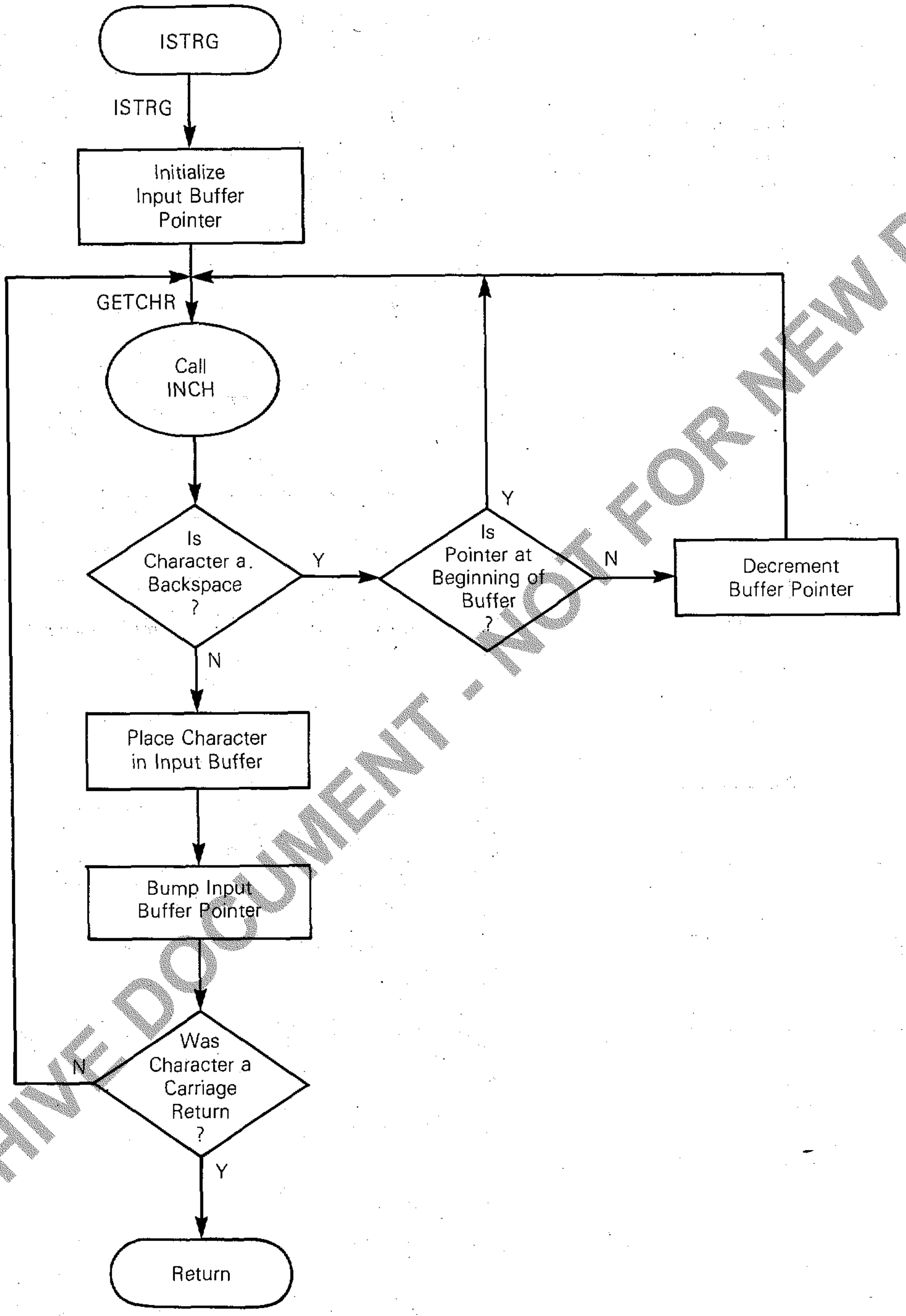

ISTRG
ISTRG
Initialize Input Buffer Pointer
GETCHR
Call INCH
Is Character a Backspace ?
Y
Is Pointer at Beginning of Buffer ?
Y
N
Decrement Buffer Pointer
N
Place Character in Input Buffer
Bump Input Buffer Pointer
Was Character a Carriage Return ?
N
Y
Return

FIGURE 3 — Dual-Tasking Software Flowcharts (Sheet 3 of 5)

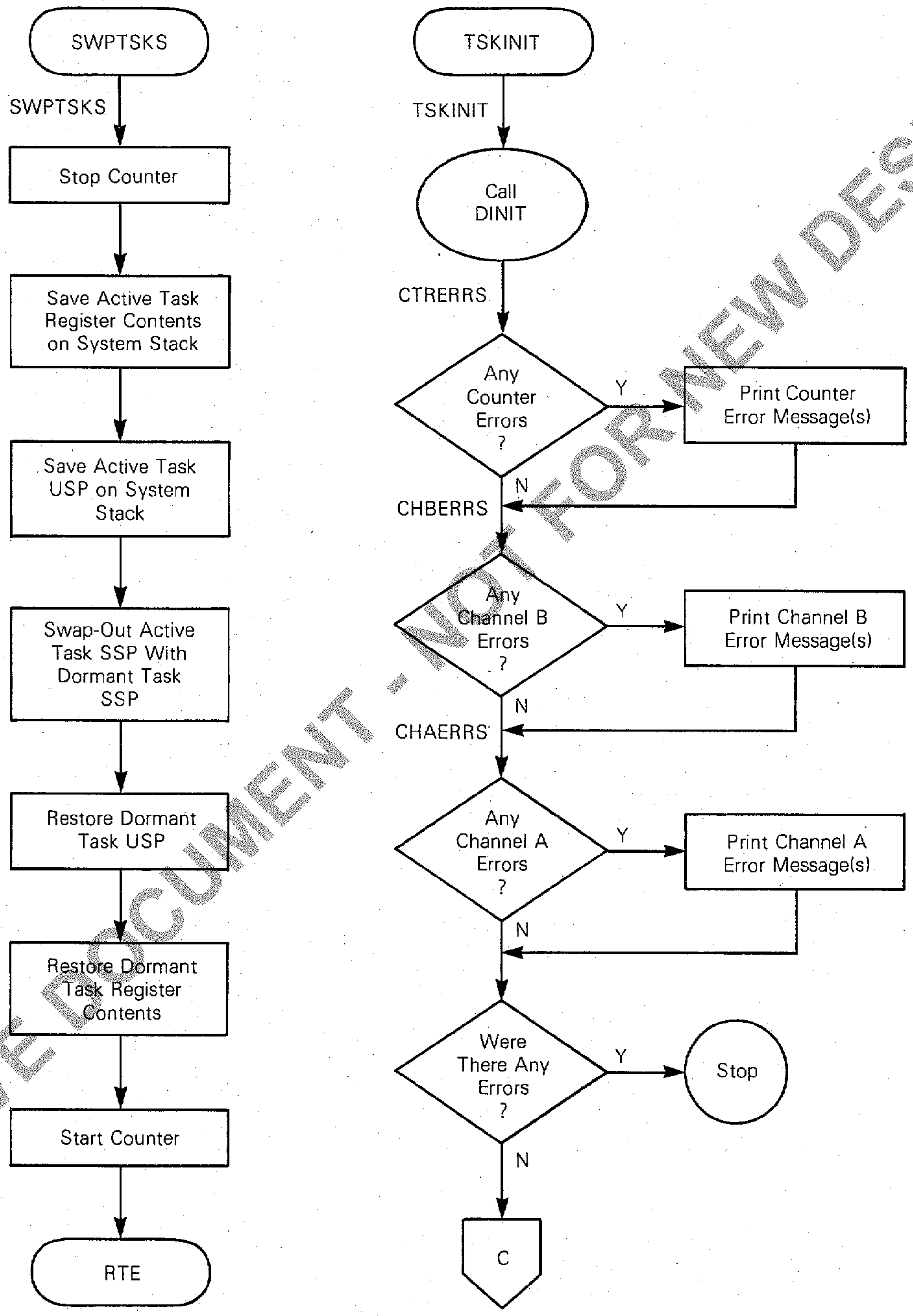

SWPTSKS
SWPTSKS
Stop Counter
Save Active Task Register Contents on System Stack
Save Active Task USP on System Stack
Swap-Out Active Task SSP With Dormant Task SSP
Restore Dormant Task USP
Restore Dormant Task Register Contents
Start Counter
RTE
TSKINIT
TSKINIT
Call DINIT
CTRERRS
Any Counter Errors ?
Y
Print Counter Error Message(s)
N
CHBERRS
Any Channel B Errors ?
Y
Print Channel B Error Message(s)
N
CHAERRS
Any Channel A Errors ?
Y
Print Channel A Error Message(s)
N
Were There Any Errors ?
Y
Stop
N
C

FIGURE 3 — Dual-Tasking Software Flowcharts (Sheet 4 of 5)

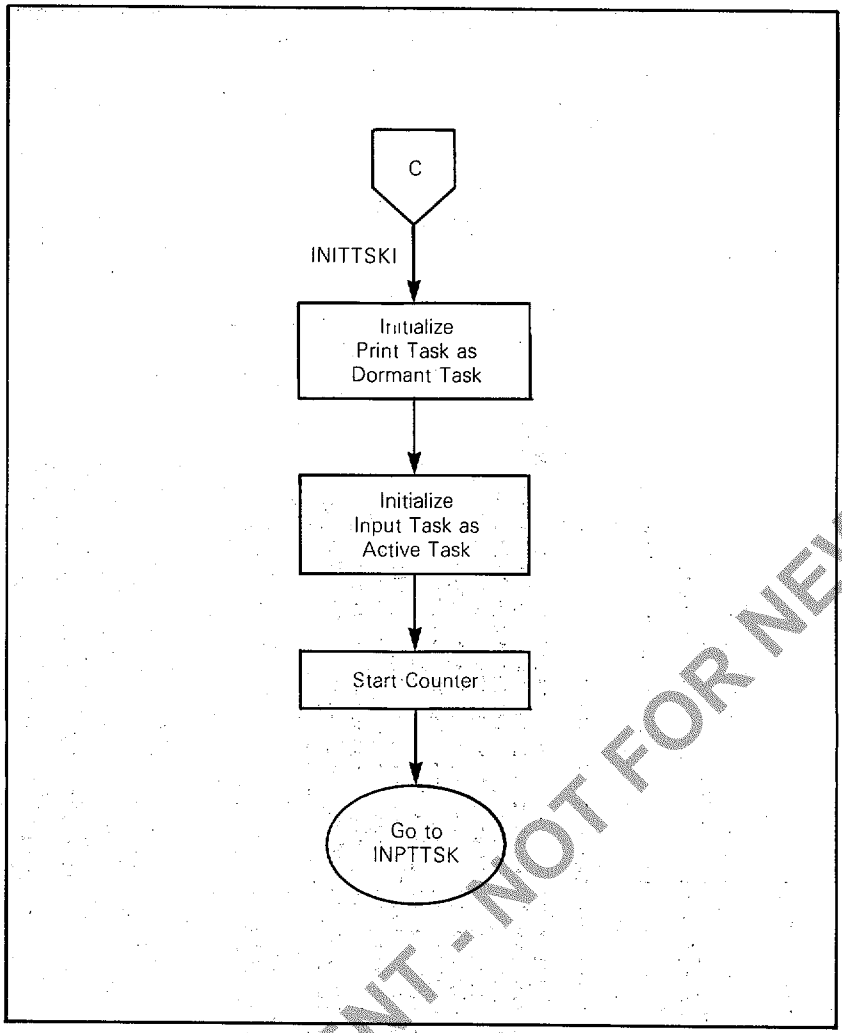

C
INITTSKI
Initialize Print Task as Dormant Task
Initialize Input Task as Active Task
Start Counter
Go to INPTTSK

FIGURE 3 — Dual-Tasking Software Flowcharts (Sheet 5 of 5)

```
MOTOROLA M68000 ASM VERSION  1.30SYS :   5.APPNOTE .DUART68S.SA 04/12/84 15:14:00                                 PAGE   1
DUART68S

                                 OPT       FRS,PCS,BRS
                           *
                           * DUART68S - ROUTINES REQUIRED FOR A 68000-BASED SYSTEM TO PERFORM
                           *            ASYNCHRONOUS SERIAL I/O & DUAL TASK EXECUTION USING
                           *            A 68681 DUART.
                           *            THE FOLLOWING ROUTINES ALLOW A 68000'S EXECUTION TIME TO
                           *            BE SPLIT BETWEEN TWO TASKS:
                           *
                           *                  TSKINIT - ROUTINE TO INITIALIZE THE TWO TASKS
                           *                  INPTTSK - SAMPLE INPUT TASK
                           *                  PRNTTSK - SAMPLE PRINT TASK
                           *                  SWPTSKS - ROUTINE TO SWAP BETWEEN TASKS
                           *
                           *            INPTTSK CONTINUALLY MONITORS A TERMINAL CONNECTED
                           *            TO THE DUART'S CHANNEL A FOR INCOMING CHARACTER STRINGS.
                           *            PRNTTSK SENDS CHARACTER STRINGS TO A PRINTER CONNECTED
                           *            TO THE DUART'S CHANNEL B.
                           *            THE TIME-SLICING INTERRUPT THAT FACILITATES THE DUAL-TASKING
                           *            IS GENERATED BY THE DUART'S COUNTER.
                           *
                           *            THE FOLLOWING ROUTINES PERFORM THE I/O OPERATIONS:
                           *
                           *                  QSTRG - SUBROUTINE TO PLACE A CHAR STRING IN PRINT QUEUE
                           *                  RSTRG - SUBROUTINE TO REMOVE A CHAR STRING FROM PRINT QUEUE
                           *                  ISTRG - SUBROUTINE TO GET A CHAR STRING FROM TERMINAL
                           *                  PSTRG - SUBROUTINE TO SEND A CHAR STRING TO PRINTER
                           *                  DINIT - SUBROUTINE TO INIT, CHECK & ENABLE DUART
                           *                  CHCHK - SUBROUTINE TO CHECK CHANNEL OPERATION
                           *                 CTRCHK - SUBROUTINE TO CHECK COUNTER OPERATION
                           *                   CIRQ - DUART INTERRUPT HANDLER USED DURING CTRCHK
                           *                   INCH - SUBROUTINE TO INPUT CHARACTER FROM TERMINAL
                           *                  OUTCH - SUBROUTINE TO OUTPUT CHARACTER TO TERMINAL
                           *                 POUTCH - SUBROUTINE TO OUTPUT CHARACTER TO PRINTER
                           *                   DIRQ - DUART INTERRUPT HANDLER
                           *
                           *            AUTHOR    - KYLE HARPER
                           *            DATE      - APRIL 9, 1984
                           *            VERSION   - 4
                           *
                           *

                           *
                           * SYSTEM ADDRESSES
                           *

          00F00001         DUART     EQU       $F00001               BASE ADDRESS OF 68681 DUART

          00F00001         CHANA     EQU       DUART+0               CHANNEL A BASE ADDRESS
          00F00001         MR1A      EQU       DUART+0               MODE REGISTER 1A
          00F00001         MR2A      EQU       DUART+0               MODE REGISTER 2A
          00F00003         SRA       EQU       DUART+2               STATUS REGISTER A
          00F00003         CSRA      EQU       DUART+2               CLOCK-SELECT REGISTER A
          00F00005         CRA       EQU       DUART+4               COMMAND REGISTER A
          00F00007         RBA       EQU       DUART+6               RECEIVER BUFFER A
          00F00007         TBA       EQU       DUART+6               TRANSMITTER BUFFER A

          00F00009         IPCR      EQU       DUART+8               INPUT PORT CHANGE REGISTER
          00F00009         ACR       EQU       DUART+8               AUXILIARY CONTROL REGISTER
```

```
            00F0000B    ISR      EQU     DUART+10          INTERRUPT STATUS REGISTER
            00F0000B    IMR      EQU     DUART+10          INTERRUPT MASK REGISTER
            00F0000D    CMSB     EQU     DUART+12          CURRENT COUNTER/TIMER MOST SIGNIFICANT BYTE
            00F0000D    CTUR     EQU     DUART+12          COUNTER/TIMER UPPER REGISTER
            00F0000F    CLSB     EQU     DUART+14          CURRENT COUNTER/TIMER LEAST SIGNIFICANT BYTE
            00F0000F    CTLR     EQU     DUART+14          COUNTER/TIMER LOWER REGISTER

            00F00011    CHANB    EQU     DUART+16          CHANNEL B BASE ADDRESS
            00F00011    MR1B     EQU     DUART+16          MODE REGISTER 1B
            00F00011    MR2B     EQU     DUART+16          MODE REGISTER 2B
            00F00013    SRB      EQU     DUART+18          STATUS REGISTER B
            00F00013    CSRB     EQU     DUART+18          CLOCK-SELECT REGISTER B
            00F00015    CRB      EQU     DUART+20          COMMAND REGISTER B
            00F00017    RBB      EQU     DUART+22          RECEIVER BUFFER B
            00F00017    TBB      EQU     DUART+22          TRANSMITTER BUFFER B

            00F00019    IVR      EQU     DUART+24          INTERRUPT VECTOR REGISTER
            00F0001B    IP       EQU     DUART+26          INPUT PORT (UNLATCHED)
            00F0001B    OPCR     EQU     DUART+26          OUTPUT PORT CONFIGURATION REGISTER
            00F0001D    STRC     EQU     DUART+28          START-COUNTER COMMAND
            00F0001D    BTST     EQU     DUART+28          OUTPUT PORT REGISTER BIT SET COMMAND
            00F0001F    STPC     EQU     DUART+30          STOP-COUNTER COMMAND
            00F0001F    BTRST    EQU     DUART+30          OUTPUT PORT REGISTER BIT RESET COMMAND

            00003800    IUSP     EQU     $003800           INPUT TASK'S USER STACK AREA
            00004000    ISSP     EQU     $004000           INPUT TASK'S SYSTEM STACK AREA
            00004800    PUSP     EQU     $004800           PRINT TASK'S USER STACK AREA
            00005000    PSSP     EQU     $005000           PRINT TASK'S SYSTEM STACK AREA

            00000000    MONITOR  EQU     $000000           MONITOR WARM-START ADDRESS

                        *
                        * CONSTANTS
                        *

            00000080    CSLNTH   EQU     128               CHARACTER STRING LENGTH IN BYTES (MAX=256)
            00000100    PQLNTH   EQU     256               PRINT QUEUE LENGTH IN BYTES (MAX=256)

            0000007F    CSLMSK   EQU     CSLNTH-1          CHARACTER STRING LENGTH MASK
            000000FF    PQLMSK   EQU     PQLNTH-1          PRINT QUEUE LENGTH MASK

            0000FFFF    TXCNT    EQU     $FFFF             TX WAIT LOOP COUNT (MAX=$FFFF)
            0000FFFF    RXCNT    EQU     $FFFF             RX WAIT LOOP COUNT (MAX=$FFFF)
            0000FFFF    IRQCNT   EQU     $FFFF             IRQ WAIT LOOP COUNT (MAX=$FFFF)

            0000000C    IRQMSK   EQU     $0C               IRQ MASK: ALLOWS CHANNEL A BREAK, & COUNTER IRQ

            0000000D    CR       EQU     $0D               ASCII CARRIAGE RETURN
            0000000A    LF       EQU     $0A               ASCII LINE FEED
            00000008    BS       EQU     $08               ASCII BACKSPACE


            00002000             ORG     $002000

                        *
                        * TSKINIT - ROUTINE TO INITIALIZE THE TWO TASKS TO BE EXECUTED BY THE 68000.
                        *           TSKINIT INITIALIZES & CHECKS THE DUART CHANNELS & COUNTER, ENABLES
                        *           THE CHANNELS, INITIALIZES THE PRINT TASK AS THE DORMANT TASK,
                             *           STARTS THE COUNTER, THEN BEGINS EXECUTION OF THE INPUT TASK.
                             *
                             *

00002000 4FEFFFFA   TSKINIT  LEA.L    -6(A7),A7             ALLOCATE STACK SPACE FOR STATUS WORDS
00002004 61000218            BSR.L    DINIT                 INITIALIZE & CHECK DUART
00002008 4C9F0007            MOVEM.W  (A7)+,D0-D2           PULL STATUS WORDS OFF STACK

0000200C 4A40       CTRERRS  TST.W    D0                    COUNTER ERROR(S)?
0000200E 670C                BEQ      CHBERRS               NO, SKIP NEXT PART
00002010 4BF8244D            LEA      CTRERR,A5             YES, PRINT COUNTER ERROR MESSAGE
00002014 4DED0023            LEA      LCTRERR(A5),A6
00002018 61000086            BSR.L    PRTMSG

0000201C 4A41       CHBERRS  TST.W    D1                    CHANNEL B ERROR(S)?
0000201E 6754                BEQ      CHAERRS               NO, SKIP NEXT PART

00002020 08010000   CHBERR1  BTST     #0,D1                 YES, IS IT TX NEVER READY?
00002024 670C                BEQ      CHBERR2               NO, SKIP NEXT PART
00002026 4BF82470            LEA      CHBMSG1,A5            YES, PRINT TX-NEVER-READY MESSAGE
0000202A 4DED0034            LEA      LCHBMSG1(A5),A6
0000202E 610000A0            BSR.L    PRTMSG

00002032 08010001   CHBERR2  BTST     #1,D1                 IS IT RX NEVER READY?
00002036 670C                BEQ      CHBERR3               NO, SKIP NEXT PART
00002038 4BF824A4            LEA      CHBMSG2,A5            YES, PRINT RX-NEVER-READY MESSAGE
0000203C 4DED002B            LEA      LCHBMSG2(A5),A6
00002040 6100008E            BSR.L    PRTMSG

00002044 08010002   CHBERR3  BTST     #2,D1                 IS IT A FRAMING ERROR?
00002048 670A                BEQ      CHBERR4               NO, SKIP NEXT PART
0000204A 4BF824CF            LEA      CHBMSG3,A5            YES, PRINT FRAMING-ERROR MESSAGE
0000204E 4DED001D            LEA      LCHBMSG3(A5),A6
00002052 617C                BSR      PRTMSG

00002054 08010003   CHBERR4  BTST     #3,D1                 IS IT A PARITY ERROR?
00002058 670A                BEQ      CHBERR5               NO, SKIP NEXT PART
0000205A 4BF824EC            LEA      CHBMSG4,A5            YES, PRINT PARITY-ERROR MESSAGE
0000205E 4DED001C            LEA      LCHBMSG4(A5),A6
00002062 616C                BSR      PRTMSG

00002064 08010004   CHBERR5  BTST     #4,D1                 IS IT A BAD CHARACTER?
00002068 670A                BEQ      CHAERRS               NO, SKIP NEXT PART
0000206A 4BF82508            LEA      CHBMSG5,A5            YES, PRINT BAD-CHARACTER MESSAGE
0000206E 4DED002C            LEA      LCHBMSG5(A5),A6
00002072 615C                BSR      PRTMSG

00002074 4A42       CHAERRS  TST.W    D2                    CHANNEL A ERROR(S)?
00002076 6750                BEQ      ERRCHK                NO, SKIP NEXT PART

00002078 08020000   CHAERR1  BTST     #0,D2                 YES, IS IT TX NEVER READY?
0000207C 670A                BEQ      CHAERR2               NO, SKIP NEXT PART
0000207E 4BF82534            LEA      CHAMSG1,A5            YES, PRINT TX-NEVER-READY MESSAGE
00002082 4DED0034            LEA      LCHAMSG1(A5),A6
00002086 6148                BSR      PRTMSG

00002088 08020001   CHAERR2  BTST     #1,D2                 IS IT RX NEVER READY?
0000208C 670A                BEQ      CHAERR3               NO, SKIP NEXT PART
```

```
0000208E 48F82568                  LEA       CHAMSG2,A5            YES, PRINT RX-NEVER-READY MESSAGE
00002092 4DED0028                  LEA       LCHAMSG2(A5),A6
00002096 6138                      BSR       PRTMSG

00002098 08020002         CHAERR3  BTST      #2,D2                 IS IT A FRAMING ERROR?
0000209C 670A                      BEQ       CHAERR4               NO, SKIP NEXT PART
0000209E 48F82593                  LEA       CHAMSG3,A5            YES, PRINT FRAMING-ERROR MESSAGE
000020A2 4DED001D                  LEA       LCHAMSG3(A5),A6
000020A6 6128                      BSR       PRTMSG

000020A8 08020003         CHAERR4  BTST      #3,D2                 IS IT A PARITY ERROR?
000020AC 670A                      BEQ       CHAERR5               NO, SKIP NEXT PART
000020AE 48F825B0                  LEA       CHAMSG4,A5            YES, PRINT PARITY-ERROR MESSAGE
000020B2 4DED001C                  LEA       LCHAMSG4(A5),A6
000020B6 6118                      BSR       PRTMSG

000020B8 08020004         CHAERR5  BTST      #4,D2                 IS IT A BAD CHARACTER?
000020BC 675C                      BEQ       INPTTSK               NO, SKIP NEXT PART
000020BE 48F825CC                  LEA       CHAMSG5,A5            YES, PRINT BAD-CHARACTER MESSAGE
000020C2 4DED002C                  LEA       LCHAMSG5(A5),A6
000020C6 6108                      BSR       PRTMSG

000020C8 8041             ERRCHK   OR.W      D1,D0                 WERE THERE ANY ERRORS?
000020CA 8042                      OR.W      D2,D0
000020CC 670A                      BEQ       INITTSK1              NO, CONTINUE WITH DEMO
000020CE 60FE                      BRA       *                     YES, STOP.

000020D0 1E3C00F3         PRTMSG   MOVE.B    #243,D7               PRINT MESSAGE TO SCREEN
000020D4 4E4E                      TRAP      #14
000020D6 4E75                      RTS


                          *
                          * INITIALIZE PRINT TASK (PRNTTSK) AS DORMANT TASK, INITIALIZE
                          * PRINT QUEUE, START COUNTER, THEN BEGIN EXECUTION OF THE INPTTSK.
                          * 68000 WILL EXECUTE INPTTSK UNTIL THE COUNTER GENERATES AN IRQ.
                          * THE 68000 WILL THEN BEGIN EXECUTING PRNTTSK AND INPTTSK WILL
                          * BECOME THE DORMANT TASK.
                          *

000020D8 2E7C00005000 INITTSK1 MOVE.L    #PSSP,A7              INIT PRINT TASK'S SYSTEM STACK POINTER
000020DE 2F3C00002122          MOVE.L    #PRNTTSK,-(A7)        INIT PRINT TASK'S PROGRAM COUNTER
000020E4 3F3C2300              MOVE.W    #$2300,-(A7)          INIT PRINT TASK'S STATUS REGISTER:IPL4-7
000020E8 700E                  MOVEQ.L   #14,D0                INIT PRINT TASK'S REGISTERS
000020EA 42A7         INITTSK2 CLR.L     -(A7)
000020EC 51C8FFFC              DBRA      D0,INITTSK2
000020F0 2F3C00004800          MOVE.L    #PUSP,-(A7)           INIT PRINT TASK'S USER STACK POINTER
000020F6 21CF7000              MOVE.L    A7,DTSKSSP            SAVE PRINT TASK'S SYSTEM STACK POINTER

000020FA 42387084              CLR.B     PQIN                  INIT PRINT QUEUE INPUT POINTER
000020FE 42387085              CLR.B     PQOUT                 INIT PRINT QUEUE OUTPUT POINTER

00002102 2E7C00003800          MOVE.L    #IUSP,A7              INIT INPUT TASK'S USER STACK POINTER
00002108 4E67                  MOVE.L    A7,USP
0000210A 2E7C00004000          MOVE.L    #ISSP,A7              INIT INPUT TASK'S SYSTEM STACK POINTER
00002110 46FC2300              MOVE.W    #$2300,SR             INIT INPUT TASK'S STATUS REGISTER:IPL4-7

00002114 4A3900F0001D          TST.B     STRC                  START COUNTER

```

```
                         *
                         * INPTTSK - TASK THAT CONTINUALLY CHECKS TERMINAL FOR INCOMING CHARACTER
                         *           STRINGS. WHEN THE COMPLETE CHARACTER STRING HAS BEEN RECEIVED,
                         *           INPTTSK SUBMITS THE STRING TO THE PRINT QUEUE.
                         *

0000211A 61000086        INPTTSK  BSR.L    ISTRG                  INPUT STRING FROM CHANNEL A
0000211E 612E                     BSR      QSTRG                  SUBMIT STRING TO PRINT QUEUE
00002120 60F8                     BRA      INPTTSK

                         *
                         * PRNTTSK - TASK THAT CONTINUALLY CHECKS PRINTER QUEUE FOR STRINGS TO BE
                         *           PRINTED. WHEN A STRING IS TO BE PRINTED, PRNTTSK WILL SEND THE
                         *           STRING FROM THE PRINT BUFFER TO THE PRINTER. IF NO STRINGS NEED
                         *           TO BE PRINTED, PRNTTSK WILL CONTINUE CHECKING QUEUE FOR STRINGS
                         *           TO BE PRINTED.
                         *

00002122 6172            PRNTTSK  BSR      RSTRG                  RELEASE STRING FROM PRINT QUEUE
00002124 60FC                     BRA      PRNTTSK                CHECK QUEUE FOR ANOTHER PRINT TAG

                         *
                         * SWPTSKS - ROUTINE TO SWAP TASKS BEING EXECUTED BY THE 68000.
                         *           SWPTSKS SWAPS BETWEEN TWO TASKS BY EXCHANGING THE
                         *           SYSTEM STACK POINTER, REGISTER CONTENTS, USER STACK POINTER,
                         *           STATUS REGISTER, & PROGRAM COUNTER OF ONE TASK TO THAT OF THE OTHER
                         *
                         *           ENTRY CONDITIONS:
                         *
                         *                  DRMNT TASK'S SSP IN DTSKSSP.
                         *                  ACTIVE TASK'S SSP IN A7.
                         *                  SSP+0 - ACTIVE TASK'S STATUS REGISTER CONTENTS.
                         *                  SSP+2 - ACTIVE TASK'S PROGRAM COUNTER CONTENTS.
                         *
                         *           EXIT CONDITIONS:
                         *
                         *                  NEW DRMNT TASK'S SSP IN DTSKSSP.
                         *                  NEW ACTIVE TASK'S SSP IN A7.
                         *                  SSP+0 - NEW ACTIVE TASK'S STATUS REGISTER CONTENTS
                         *                  SSP+2 - NEW ACTIVE TASK'S PROGRAM COUNTER CONTENTS
                         *
                         *

00002126 4A3900F0001F    SWPTSKS  TST.B    STPC                   STOP COUNTER

0000212C 48E7FFFE                 MOVEM.L  A0-A6/D0-D7,-(A7)      SAVE ACTIVE TASK'S REGISTER CONTENTS
00002130 4E6E                     MOVE.L   USP,A6                 SAVE ACTIVE TASK'S USER STACK POINTER
00002132 2F0E                     MOVE.L   A6,-(A7)

00002134 4DD7                     LEA.L    (A7),A6                SAVE TEMP COPY OF ACTIVE TASK'S SSP
00002136 2E787000                 MOVE.L   DTSKSSP,A7             GET DRMNT TASK'S SYSTEM STACK POINTER
0000213A 21CE7000                 MOVE.L   A6,DTSKSSP             SAVE ACTIVE TASK'S SYSTEM STACK POINTER

0000213E 2C5F                     MOVE.L   (A7)+,A6               GET DRMNT TASK'S USER STACK POINTER
00002140 4E66                     MOVE.L   A6,USP
00002142 4CDF7FFF                 MOVEM.L  (A7)+,D0-D7/A0-A6      GET DRMNT TASK'S REGISTER CONTENTS

00002146 4A3900F0001D             TST.B    STRC                   START COUNTER
```

```
0000214C 4E73                   RTE                               RETURN FROM EXCEPTION TO NEW ACTIVE TASK

                        *
                        * QSTRG  - SUBROUTINE TO SUBMIT A CHARACTER STRING TO PRINT QUEUE.
                        *          QSTRG CHECKS THE STATUS OF THE PRINT QUEUE. IF IT IS
                        *          FULL, QSTRG WILL WAIT UNTIL THERE IS ROOM IN THE QUEUE FOR
                        *          A TAG. IF THE QUEUE IS NOT FULL, QSTRG WILL MOVE THE CHARACTER
                        *          STRING INTO THE PRINT BUFFER, & PLACE A PRINT TAG IN THE PRINT
                        *          QUEUE.
                        *          A PRINT TAG IS A BYTE CONTAINING THE LENGTH OF THE STRING TO BE
                        *          PRINTED.
                        *
                        *          ENTRY CONDITIONS:
                        *
                        *                  A0 CONTAINS STRING'S START ADDRESS.
                        *                  D1 CONTAINS STRING'S LENGTH (MAX = 256 CHARACTERS).
                        *
                        *          EXIT CONDITIONS:
                        *
                        *                  CHARACTER STRING MOVED INTO PRINT BUFFER.
                        *                  PRINT TAG PLACED IN PRINT QUEUE.
                        *                  ALL REGISTERS UNALTERED.
                        *
                        *

0000214E 48E7F0C0       QSTRG    MOVEM.L  A0-A1/D0-D3,-(A7)   SUBROUTINE USES REGS A0,A1,D2-D4

00002152 4242                    CLR.W    D2                  GET PRINT QUEUE INPUT POINTER
00002154 14387084                MOVE.B   PQIN,D2
00002158 5202                    ADDQ.B   #1,D2               BUMP INPUT POINTER
0000215A 020200FF                ANDI.B   #PQLMSK,D2          (KEEP POINTER WITHIN QUEUE BOUNDS)
0000215E B4387085       QSTRG1   CMP.B    PQOUT,D2            IS PRINT QUEUE FULL (PQIN+1=PQOUT)?
00002162 67FA                    BEQ      QSTRG1              YES, WAIT UNTIL HAVE ROOM FOR TAG

00002164 43F87186                LEA.L    PRTBUF,A1           NO, MOVE STRING INTO PRINT BUFFER:
00002168 4283                    CLR.L    D3                  GET STRING DESTINATION ADDRESS BY
0000216A 3602                    MOVE.W   D2,D3               ADDING INPUT OFFSET (PQIN * CSLNTH)
0000216C C6FC0080                MULU.W   #CSLNTH,D3          TO
00002170 43F13800                LEA      0(A1,D3.L),A1       PRINT BUFFER BASE ADDRESS

00002174 4240                    CLR.W    D0                  GET STRING LENGTH
00002176 1001                    MOVE.B   D1,D0
00002178 5300                    SUBQ.B   #1,D0               DECREMENT IT BY 1
0000217A 0200007F                ANDI.B   #CSLMSK,D0          (KEEP IT WITHIN STRING LENGTH BOUNDS)

0000217E 12D8           QSTRG2   MOVE.B   (A0)+,(A1)+         MOVE STRING
00002180 51C8FFFC                DBRA     D0,QSTRG2

00002184 43F87086                LEA.L    PQUE,A1             PLACE PRINT TAG IN PRINT QUEUE
00002188 13812000                MOVE.B   D1,0(A1,D2.W)

0000218C 11C27084                MOVE.B   D2,PQIN             UPDATE PRINT QUEUE INPUT POINTER

00002190 4CDF030F                MOVEM.L  (A7)+,A0-A1/D0-D3   RESTORE REGISTER CONTENTS
00002194 4E75                    RTS

                        *
                        * RSTRG - SUBROUTINE TO RELEASE A CHARACTER STRING FROM PRINT QUEUE.
```

```
                       *         RSTRG CHECKS THE STATUS OF THE PRINT QUEUE. IF THE QUEUE IS
                       *         EMPTY, RSTRG WILL WAIT UNTIL A PRINT TAG APPEARS IN THE QUEUE.
                       *         A PRINT TAG IS A BYTE CONTAINING THE LENGTH OF THE STRING TO
                       *         BE PRINTED.
                       *         IF THE PRINT QUEUE IS NOT EMPTY, RSTRG WILL SEND THE STRING
                       *         FROM THE PRINT BUFFER TO THE PRINTER, THEN PULL THE TAG FROM THE
                       *         PRINT QUEUE.
                       *
                       *        ENTRY CONDITIONS:
                       *
                       *                  (NONE)
                       *
                       *        EXIT CONDITIONS:
                       *
                       *                  CHARACTER STRING IS SENT FROM THE PRINT BUFFER
                       *                  TO CHANNEL B.
                       *                  PRINT TAG IS REMOVED FROM PRINT QUEUE.
                       *                  ALL REGISTERS UNALTERED.
                       *
                       *

00002196 48E7C0C0      RSTRG    MOVEM.L  D0-D1/A0-A1,-(A7)      SUBROUTINE USES REGS D0, D1, A0, & A1

0000219A 4240                   CLR.W    D0                     GET PRINT QUEUE OUTPUT POINTER
0000219C 10387085               MOVE.B   PQOUT,D0
000021A0 B0387084      RSTRG1   CMP.B    PQIN,D0                IS PRINT QUEUE EMPTY (PQOUT=PQIN)?
000021A4 67FA                   BEQ      RSTRG1                 YES, WAIT FOR A TAG TO APPEAR IN QUEUE

000021A6 41F87186               LEA.L    PRTBUF,A0              NO, RELEASE STRING:
000021AA 4281                   CLR.L    D1                     GET STRING SOURCE ADDRESS BY
000021AC 3200                   MOVE.W   D0,D1                  ADDING OUTPUT OFFSET (PQOUT * CSLNTH)
000021AE C2FC0080               MULU.W   #CSLNTH,D1             TO
000021B2 41F01800               LEA.L    0(A0,D1.L),A0          PRINT BUFFER BASE ADDRESS

000021B6 43F87086               LEA.L    PQUE,A1                GET STRING LENGTH
000021BA 4241                   CLR.W    D1                     FROM
000021BC 12310000               MOVE.B   0(A1,D0.W),D1          PRINT TAG

000021C0 6142                   BSR      PSTRG                  SEND STRING TO CHANNEL B

000021C2 5200                   ADDQ.B   #1,D0                  BUMP PRINT QUEUE OUTPUT POINTER
000021C4 020000FF               ANDI.B   #PQLMSK,D0             (KEEP POINTER WITHIN QUEUE BOUNDS)
000021C8 11C07085               MOVE.B   D0,PQOUT               UPDATE PRINT QUEUE OUTPUT POINTER

000021CC 4CDF0303               MOVEM.L  (A7)+,D0-D1/A0-A1      RESTORE REGISTER CONTENTS
000021D0 4E75                   RTS

                       *
                       * ISTRG  - ROUTINE TO INPUT A CHARACTER STRING FROM THE TERMINAL & PLACE
                       *          IT IN INPUT BUFFER.
                       *          A CHARACTER STRING CAN BE A MAXIMUM OF 256 CHARACTERS LONG
                       *          (AS DEFINED BY THE CSLNTH), & ENDS WITH CARRIAGE RETURN CHARACTER.
                       *          IF A BACKSPACE IS RECEIVED, ISTRG WILL DECREMENT THE INPUT
                       *          BUFFER POINTER UNLESS POINTER IS AT FIRST POSITION IN BUFFER.
                       *
                       *          ENTRY CONDITIONS:
                       *
                       *                  (NONE)
```

```
                              *
                              *         EXIT CONDITIONS:
                              *
                              *                 CHARACTER STRING IS IN INPUT BUFFER.
                              *                 A0 CONTAINS START ADDRESS OF INPUT BUFFER.
                              *                 D1 CONTAINS LENGTH OF STRING.
                              *                 ALL OTHER REGISTERS ARE RESTORED.
                              *
                              *

00002102 48E78000             ISTRG     MOVEM.L  D0,-(A7)              SUBROUTINE USES REGISTERS D0

000021D6 41F87004                       LEA.L    INBUF,A0              GET BASE ADDRESS OF INPUT BUFFER
000021DA 4241                           CLR.W    D1                    INIT INPUT BUFFER POINTER
000021DC 610001B2             GETCHAR   BSR.L    INCH                  GET CHARACTER FROM CHANNEL A

000021E0 0C000008             BSCHK     CMP.B    #BS,D0                IS IT A BACKSPACE CHARACTER?
000021E4 6608                           BNE      PUTCHAR               NO, SKIP NEXT PART
000021E6 4A01                           TST.B    D1                    YES, ARE WE AT BEGINNING OF BUFFER?
000021E8 67F2                           BEQ      GETCHAR               YES, DO NOT DECREMENT POINTER
000021EA 5301                           SUBQ.B   #1,D1                 NO, DECREMENT BUFFER POINTER
000021EC 60EE                           BRA      GETCHAR               THEN GET NEXT CHARACTER

000021EE 11801000             PUTCHAR   MOVE.B   D0,0(A0,D1.W)         PUT CHARACTER IN INPUT BUFFER,
000021F2 5201                           ADDQ.B   #1,D1                 BUMP BUFFER POINTER
000021F4 0201007F                       ANDI.B   #CSLMSK,D1            (KEEP IT WITHIN STRING LENGTH BOUNDS)
000021F8 0C00000D                       CMP.B    #CR,D0                WAS IT A CARRIAGE RETURN?
000021FC 66DE                           BNE      GETCHAR               NO, GET NEXT CHAR

000021FE 4CDF0001                       MOVEM.L  (A7)+,D0              YES, RESTORE REGISTER CONTENTS & RETURN
00002202 4E75                           RTS

                              *
                              * PSTRG   - ROUTINE TO SEND A CHARACTER STRING TO THE PRINTER.
                              *
                              *         ENTRY CONDITIONS:
                              *
                              *                 A0 CONTAINS STRING'S START ADDRESS.
                              *                 D1 CONTAINS STRING'S LENGTH (MAX = 256 CHARACTERS).
                              *
                              *         EXIT CONDITIONS:
                              *
                              *                 CHARACTER STRING IS SENT TO PRINTER VIA CHANNEL B.
                              *                 ALL REGISTERS ARE UNALTERED.
                              *
                              *

00002204 48E7C080             PSTRG     MOVEM.L  A0/D0-D1,-(A7)        SUBROUTINE USES REGS A0,D0,D1

00002208 5301                           SUBQ.B   #1,D1                 INIT CHARACTER COUNT FROM STRING LENGTH
0000220A 0201007F                       ANDI.B   #CSLMSK,D1            (KEEP IT WITHIN STRING LENGTH BOUNDS)
0000220E 1018                 PSTRG1    MOVE.B   (A0)+,D0              GET CHAR OF STRING TO BE PRINTED
00002210 610001BA                       BSR.L    POUTCH                PRINT CHARACTER
00002214 51C9FFF8                       DBRA     D1,PSTRG1             WAS IT THE LAST CHARACTER OF STRING?

00002218 4CDF0103                       MOVEM.L  (A7)+,A0/D0-D1        YES, RESTORE REGISTER CONTENTS
0000221C 4E75                           RTS

```

```
                      *
                      * DINIT - DUART INITIALIZATION ROUTINE.
                      *         AFTER INITIALIZING THE DUART'S CHANNELS & COUNTER FOR
                      *         OPERATION, DINIT CHECKS CHANNEL A, CHANNEL B, & THE
                      *         COUNTER FOR OPERATIONAL ERRORS.
                      *
                      *         ENTRY CONDITIONS:
                      *
                      *                  ALLOCATE THREE WORDS ON SYSTEM STACK BEFORE CALLING.
                      *
                      *         EXIT CONDITIONS:
                      *
                      *                  THREE STATUS WORDS ARE PLACED ON THE SYSTEM STACK.
                      *
                      *                  THE STATUS WORDS' FORMATS ARE AS FOLLOWS:
                      *
                      *                   WORD     BIT       STATUS (1=ERROR, 0=NO ERROR)
                      *                   ----     ---       ----------------------------
                      *
                      *                  (A7)+0     0        CHAN A TRANSMITTER NEVER READY
                      *                     "       1          "  " RECEIVER NEVER READY
                      *                     "       2          "  " FRAMING ERROR
                      *                     "       3          "  " PARITY ERROR
                      *                     "       4          "  " INCORRECT CHARACTER RECEIVED
                      *                     "      5-15      (NOT USED)
                      *
                      *                  (A7)+2     0        CHAN B TRANSMITTER NEVER READY
                      *                     "       1          "  " RECEIVER NEVER READY
                      *                     "       2          "  " FRAMING ERROR
                      *                     "       3          "  " PARITY ERROR
                      *                     "       4          "  " INCORRECT CHARACTER RECEIVED
                      *                     "      5-15      (NOT USED)
                      *
                      *                  (A7)+4     0        COUNTER IRQ NEVER RECEIVED
                      *                     "      1-15      (NOT USED)
                      *
                      *         IF NO ERRORS ARE FOUND IN CHAN A, DINIT WILL ENABLE A'S RX.
                      *         IF NO ERRORS ARE FOUND IN CHAN B, DINIT WILL ENABLE B'S TX.
                      *         THE COUNTER WILL NOT BE RUNNING.
                      *         ALL REGISTER CONTENTS ARE UNALTERED.
                      *
                      *

                      * CONSTANTS

         0000000C     CHASTS   EQU      12                     STACK OFFSET TO CHAN A STATUS WORD
         0000000E     CHBSTS   EQU      14                     STACK OFFSET TO CHAN B STATUS WORD
         00000010     CTRSTS   EQU      16                     STACK OFFSET TO COUNTER STATUS WORD

0000221E 48E78080     DINIT    MOVEM.L  A0/D0,-(A7)            SUBROUTINE USES REGS A0-A4 & D0

                      * INITIALIZE DUART CHANNELS & COUNTER

00002222 13FC003000F0          MOVE.B   #$30,ACR               BRG SET 1, CNTR MODE, CLK SRCE: X1/16
         0009
0000222A 13FC00BB00F0          MOVE.B   #$BB,CSRA              A: RX & TX AT 9600 BAUD
         0003
00002232 13FC008A00F0          MOVE.B   #$8A,MR1A              *  RX-RTS, CHAR ERR, FRCE PAR, 7 CHAR
```

```
                0001
0000223A 13FC004F00F0          MOVE.B   #$4F,MR2A            *  A-ECHO, NO TX-RTS, NO CTS-TX, 2 STOPS
         0001
00002242 13FC004400F0          MOVE.B   #$44,CSRB            B: RX & TX AT 300 BAUD
         0013
0000224A 13FC000A00F0          MOVE.B   #$0A,MR1B            *  NO RX-RTS, CHAR ERR, FRCE PAR, 7 CHAR
         0011
00002252 13FC001700F0          MOVE.B   #$17,MR2B            *  NORMAL, NO TX-RTS, CTS-TX, 1 STOP
         0011
0000225A 13FC00FF00F0          MOVE.B   #255,IVR             INIT IVR WITH IRQ VECTOR NUMBER
         0019
00002262 13FC000000F0          MOVE.B   #$00,CTUR            INIT COUNTER/TIMER REGISTERS
         0000
0000226A 13FC007300F0          MOVE.B   #$73,CTLR
         000F
00002272 13FC000C00F0          MOVE.B   #IRQMSK,IMR          INIT IRQ MASK REGISTER
         0008

                      * CHECK CHANNEL A FOR OPERATIONAL ERRORS

0000227A 41F900F00001 CHKA     LEA.L    CHANA,A0             LOAD CHANNEL A ADDRESS FOR CHECK
00002280 6142                  BSR      CHCHK                CHECK CHANNEL A
00002282 3F40000C              MOVE.W   D0,CHASTS(A7)        PLACE CHAN A STATUS WORD IN STACK

                      * CHECK CHANNEL B FOR OPERATIONAL ERRORS

00002286 41F900F00011 CHKB     LEA.L    CHANB,A0             LOAD CHANNEL B ADDRESS FOR CHECK
0000228C 6136                  BSR      CHCHK                CHECK CHANNEL B
0000228E 3F40000E              MOVE.W   D0,CHBSTS(A7)        PLACE CHAN B STATUS WORD IN STACK

                      * CHECK COUNTER FOR OPERATIONAL ERRORS

00002292 610000AC     CHKCTR   BSR.L    CTRCHK               CHECK COUNTER
00002296 3F400010              MOVE.W   D0,CTRSTS(A7)        PLACE COUNTER STATUS WORD IN STACK

                      * DUART CHECK COMPLETE, ENABLE CHANNELS UNLESS ERRORS WERE FOUND,
                      * THEN RETURN TO CALLING ROUTINE.

0000229A 4A6F000C     ENABLA   TST.W    CHASTS(A7)           ARE THERE ERRORS IN CHANNEL A?
0000229E 6610                  BNE      ENABLB               YES, SKIP NEXT PART
000022A0 13FC000100F0          MOVE.B   #$01,CRA             NO, ENABLE A'S RX,
         0005
000022A8 13FC000100F0          MOVE.B   #$01,BTST            ASSERT A'S RTS OUTPUT
         001D
000022B0 4A6F000E     ENABLB   TST.W    CHBSTS(A7)           ARE THERE ERRORS IN CHANNEL B?
000022B4 6608                  BNE      DINITR               YES, SKIP NEXT PART
000022B6 13FC000400F0          MOVE.B   #$04,CRB             NO, ENABLE B'S TX
         0015

000022BE 4CDF0101     DINITR   MOVEM.L  (A7)+,D0/A0          RESTORE REGISTER CONTENTS
000022C2 4E75                  RTS

                      *
                      * CHCHK - CHANNEL CHECK ROUTINE.
                      *         CHECKS A 68681 DUART CHANNEL FOR OPERATIONAL ERRORS.
                      *         AFTER PLACING CHANNEL IN LOCAL LOOPBACK MODE, CHCHK
                      *         CHECKS FOR THE FOLLOWING CHANNEL ERRORS:
                      *
```

```
                            *                 TRANSMITTER NEVER READY
                            *                 RECEIVER NEVER READY
                            *                 FRAMING ERROR
                            *                 PARITY ERROR
                            *                 INCORRECT CHARACTER RECEIVED
                            *
                            *         ENTRY CONDITIONS:
                            *
                            *                 CHANNEL IS ALREADY CONFIGURED FOR OPERATION, BUT NOT ENABLED
                            *                 A0 CONTAINS BASE ADDRESS OF DUART CHANNEL.
                            *
                            *         EXIT CONDITIONS:
                            *
                            *                 CHANNEL IS RESTORED TO ORIGINAL OPERATING MODE.
                            *                 A CHANNEL STATUS WORD IS PLACED IN REGISTER D0.
                            *
                            *                 THE CHANNEL STATUS WORD FORMAT IS AS FOLLOWS:
                            *
                            *                           BIT       STATUS (1=ERROR, 0=NO ERROR)
                            *                           ---       ----------------------------
                            *
                            *                            0        TRANSMITTER NEVER READY
                            *                            1        RECEIVER NEVER READY
                            *                            2        FRAMING ERROR
                            *                            3        PARITY ERROR
                            *                            4        INCORRECT CHARACTER RECEIVED
                            *                           5-15      (NOT USED)
                            *
                            *                 ALL OTHER REGISTERS ARE UNALTERED.
                            *
                            *


000022C4 48E77000           CHCHK     MOVEM.L  D1-D3,-(A7)           SUBROUTINE USES REGS D1-D3

                            * CHANGE ORIGINAL CHANNEL MODE TO LOCAL LOOPBACK MODE & CLEAR STATUS WORD

000022C8 1610                         MOVE.B   (A0),D3               SAVE ORIGINAL MR2x REGISTER CONTENTS
000022CA 00100080                     ORI.B    #$80,(A0)             PUT CHANNEL IN LOCAL LOOPBACK MODE &
000022CE 021000AF                     ANDI.B   #$AF,(A0)             MAKE SURE CTS-TX IS DISABLED FOR CHECK
000022D2 117C00050004                 MOVE.B   #$05,4(A0)            ENABLE CHANNEL'S TX
000022D8 4240                         CLR.W    D0                    CLEAR CHANNEL STATUS WORD

                            * CHECK CHANNEL'S TRANSMITTER

000022DA 323CFFFF                     MOVE.W   #TXCNT,D1             INIT TX WAIT LOOP COUNT
000022DE 082800020002 TXCHK           BTST.B   #2,2(A0)              WAIT FOR TX TO BECOME READY
000022E4 56C9FFF8                     DBNE     D1,TXCHK              WAITED TOO LONG?
000022E8 6606                         BNE      SNDCHR                NO, SKIP NEXT PART
000022EA 00400001                     ORI.W    #$0001,D0             YES, SET TX-NEVER-READY FLAG BIT
000022EE 6042                         BRA      RSTCHN                & SKIP REST OF CHECK
000022F0 117C00550006 SNDCHR          MOVE.B   #$55,6(A0)            TX IS READY, SEND TEST CHARACTER

                            * CHECK CHANNEL'S RECEIVER

000022F6 323CFFFF                     MOVE.W   #RXCNT,D1             INIT RX WAIT LOOP COUNT
000022FA 082800000002 RXCHK           BTST.B   #0,2(A0)              WAIT FOR RX TO RECEIVE CHARACTER
00002300 56C9FFF8                     DBNE     D1,RXCHK              WAITED TOO LONG?
```

```
 00002304 6606                     BNE      FRCHK                  NO, SKIP NEXT PART
 00002306 00400002                 ORI.W    #$0002,D0              YES, SET RX-NEVER-READY FLAG BIT
 0000230A 6026                     BRA      RSTCHN                 & SKIP REST OF CHECK
 0000230C 082800060002 FRCHK       BTST.B   #6,2(A0)               RX HAS CHAR, HAVE FRAMING ERROR?
 00002312 6704                     BEQ      PRCHK                  NO, SKIP NEXT PART
 00002314 00400004                 ORI.W    #$0004,D0              YES, SET FRAMING ERROR FLAG BIT
 00002318 082800050002 PRCHK       BTST.B   #5,2(A0)               HAVE PARITY PARITY ERROR?
 0000231E 6704                     BEQ      CHRCHK                 NO, SKIP NEXT PART
 00002320 00400008                 ORI.W    #$0008,D0              YES, SET PARITY ERROR FLAG BIT
 00002324 14280006     CHRCHK      MOVE.B   6(A0),D2               NO STATUS ERRORS, GET CHAR FROM RX
 00002328 0C020055                 CMP.B    #$55,D2                IS IT THE SAME CHAR TX'D?
 0000232C 6704                     BEQ      RSTCHN                 YES, SKIP NEXT PART
 0000232E 00400010                 ORI.W    #$0010,D0              NO, SET INCORRECT-CHAR-RX'D FLAG BIT

                       * CHANNEL CHECK COMPLETE, STACK STATUS WORD & RESTORE
                       * CHANNEL TO ORIGINAL MODE OF OPERATION.

 00002332 117C000A0004 RSTCHN      MOVE.B   #$0A,4(A0)             DISABLE CHANNEL'S TX
 00002338 1083                     MOVE.B   D3,(A0)                RESTORE CHANNEL TO ORIGINAL MODE

 0000233A 4CDF000E                 MOVEM.L  (A7)+,D1-D3            RESTORE REGISTER CONTENTS
 0000233E 4E75                     RTS

                       *
                       * CTRCHK - COUNTER CHECK ROUTINE.
                       *          CHECKS DUART COUNTER FOR OPERATIONAL ERRORS.
                       *          AFTER RE-POINTING THE DUART'S EXCEPTION VECTOR
                       *          TO ITS OWN INTERRUPT HANDLER, CTRCHK STARTS THE
                       *          COUNTER & WAITS FOR THE COUNTER TO GENERATE AN IRQ.
                       *
                       *          ENTRY CONDITIONS:
                       *
                       *                  DUART CONFIGURED FOR A COUNTER IRQ (IMR[3]=1).
                       *                  IRQ VECTOR REGISTER IS ALREADY INITIALIZED.
                       *                  COUNTER UPPER & LOWER REGISTERS ARE ALREADY INITIALIZED.
                       *                  COUNTER IS NOT RUNNING.
                       *
                       *          EXIT CONDITIONS:
                       *
                       *                  ORIGINAL DUART EXCEPTION VECTOR IS RESTORED.
                       *                  A COUNTER STATUS WORD IS PLACED IN REGISTER D0.
                       *
                       *                  THE ERROR STATUS WORD FORMAT IS AS FOLLOWS:
                       *
                       *                            BIT       STATUS (1=ERROR, 0=NO ERROR)
                       *                            ---       ----------------------------
                       *
                       *                             0        COUNTER IRQ NEVER RECEIVED
                       *                            1-15      (NOT USED)
                       *
                       *                  ALL OTHER REGISTERS ARE UNALTERED.
                       *
                       *


 00002340 48E74000     CTRCHK      MOVEM.L  D1,-(A7)               SUBROUTINE USES REG D1

 00002344 2F3803FC                 MOVE.L   DIRQVEC,-(A7)          SAVE ORIGINAL EXCEPTION VECTOR
```

```
00002348 21FC0000237A         MOVE.L   #CIRQ,DIRQVEC         RE-POINT EXCEPTION VECTOR
         03FC

00002350 4240                 CLR.W    D0                    CLEAR COUNTER STATUS WORD
00002352 4A3900F0001D         TST.B    STRC                  START COUNTER

00002358 323CFFFF             MOVE.W   #IRQCNT,D1            INIT IRQ WAIT LOOP COUNT
0000235C 023C00FE             ANDI.B   #$FE,CCR              CLEAR CARRY BIT

00002360 55C9FFFE     WTIRQ   DBCS     D1,WTIRQ              WAIT FOR COUNTER IRQ: WAITED TOO LONG?

00002364 6504                 BCS      CTRCHKR               NO, SKIP NEXT PART
00002366 00400001             ORI.W    #$01,D0               YES, SET IRQ-NEVER-REC'D FLAG BIT

                      * COUNTER CHECK COMPLETE, STOP COUNTER, RESTORE ORIGINAL EXCEPTION VECTOR,
                      * & STACK ERROR STATUS WORD.

0000236A 4A3900F0001F CTRCHKR TST.B    STPC                  STOP COUNTER
00002370 21DF03FC             MOVE.L   (A7)+,DIRQVEC         RESTORE ORIGINAL EXCEPTION VECTOR

00002374 4CDF0002             MOVEM.L  (A7)+,D1              RESTORE REGISTER CONTENTS
00002378 4E75                 RTS

                      *
                      * CIRQ - COUNTER CHECK IRQ HANDLING ROUTINE.
                      *        DUART IRQ HANDLING ROUTINE USED DURING CTRCHK ONLY.
                      *
                      *           ENTRY CONDITIONS:
                      *
                      *                  DUART IRQ.
                      *
                      *           EXIT CONDITIONS:
                      *
                      *                  IF COUNTER WAS CAUSE OF DUART IRQ:
                      *                            COUNTER/TIMER READY BIT CLEARED IN DUART'S ISR,
                      *                            & CARRY BIT SET.
                      *                  OTHERWISE:
                      *                            CARRY BIT REMAINS CLEARED.
                      *
                      *
0000237A 0839000300F0 CIRQ    BTST.B   #3,ISR                WAS IRQ CAUSED BY COUNTER?
         000B
00002382 670A                 BEQ      CIRQR                 NO, SKIP NEXT PART
00002384 4A3900F0001F         TST.B    STPC                  YES, STOP COUNTER
0000238A 00570001             ORI      #$0001,(A7)           & SET CARRY BIT OF SR ON STACK
0000238E 4E73         CIRQR   RTE

                      *
                      * INCH - TERMINAL INPUT CHARACTER ROUTINE.
                      *        GETS CHARACTER FROM TERMINAL VIA DUART CHANNEL A,
                      *        THEN PLACES IT IN D0.
                      *        (BECAUSE CHAN A IS IN AUTO-ECHO MODE, CHARACTER DOES NOT NEED TO
                      *        BE RE-TRANSMITTED BACK TO TERMINAL BY SOFTWARE.)
                      *
                      *        ENTRY CONDITIONS:
                      *
                      *                  DUART CHANNEL A RX & TX ENABLED.
                      *
                      *        EXIT CONDITIONS:
                      *
                      *                 RECEIVED CHARACTER PLACED IN D0.
                      *                 ALL OTHER REGISTERS UNALTERED.
                      *
                      *

00002390 0839000000F0 INCH     BTST.B   #0,SRA                 WAIT FOR CHAN A'S RX TO GET A CHAR
         0003
00002398 67F6                  BEQ      INCH
0000239A 103900F00007          MOVE.B   RBA,D0                 GET CHARACTER FROM RECEIVER
000023A0 4E75                  RTS

                      *
                      * OUTCH - TERMINAL OUTPUT CHARACTER ROUTINE.
                      *         OUTPUTS CHARACTER IN D0 TO TERMINAL VIA CHAN A'S TX.
                      *         IF CHARACTER IN D0 IS A CARRIAGE RETURN, OUTCH WILL
                      *         OUTPUT BOTH A CARRIAGE RETURN & LINE FEED CHARACTER.
                      *
                      *         ENTRY CONDITIONS:
                      *
                      *                 DUART CHANNEL A TX ENABLED.
                      *                 CHARACTER TO BE TRANSMITTED IN D0.
                      *
                      *         EXIT CONDITIONS:
                      *
                      *                 ALL REGISTERS UNALTERED.
                      *                 CHARACTER SENT TO CHANNEL A TX.
                      *
                      *

000023A2 0839000200F0 OUTCH    BTST.B   #2,SRA                 WAIT FOR CHAN A'S TX TO BECOME READY
         0003
000023AA 67F6                  BEQ      OUTCH
000023AC 13C000F00007          MOVE.B   D0,TBA                 SEND CHAR TO TRANSMITTER
000023B2 0C00000D              CMP.B    #CR,D0                 WAS IT A CARRIAGE RETURN?
000023B6 6612                  BNE      OUTCHR                 NO, SKIP NEXT PART
000023B8 0839000200F0 OUTCH1   BTST.B   #2,SRA                 YES, WAIT FOR TX TO BECOME READY AGAIN
         0003
000023C0 67F6                  BEQ      OUTCH1
000023C2 13FC000A00F0          MOVE.B   #LF,TBA                SEND A LINE FEED
         0007
000023CA 4E75         OUTCHR   RTS

                      * POUTCH - PRINTER OUTPUT CHARACTER ROUTINE.
                      *          OUTPUTS CHARACTER IN D0 TO PRINTER VIA CHAN B'S TX.
                      *          IF CHARACTER IN D0 IS A CARRIAGE RETURN, POUTCH WILL
                      *          OUTPUT BOTH A CARRIAGE RETURN & LINE FEED CHARACTER.
                      *
                      *         ENTRY CONDITIONS:
                      *
                      *                 DUART CHANNEL B TX ENABLED.
                      *                 CHARACTER TO BE TRANSMITTED IN D0.
                      *
                      *         EXIT CONDITIONS:
                      *
                      *                 ALL REGISTERS UNALTERED.
                      *                 CHARACTER SENT TO CHANNEL B TX.
```

```
                               *
                               *

000023CC 0839000200F0 POUTCH   BTST.B   #2,SRB                 WAIT FOR CHAN B'S TX TO BECOME READY
         0013
000023D4 67F6                  BEQ      POUTCH
000023D6 13C000F00017          MOVE.B   D0,TBB                 SEND CHAR TO TRANSMITTER
000023DC 0C00000D              CMP.B    #CR,D0                 WAS IT A CARRIAGE RETURN?
000023E0 6612                  BNE      POUTCHR                NO, SKIP NEXT PART
000023E2 0839000200F0 POUTCH1  BTST.B   #2,SRB                 YES, WAIT FOR TX TO BECOME READY AGAIN
         0013
000023EA 67F6                  BEQ      POUTCH1
000023EC 13FC000A00F0          MOVE.B   #LF,TBB                SEND LINE FEED TO TRANSMITTER
         0017

000023F4 4E75         POUTCHR  RTS

                      *
                      * DIRQ - DUART IRQ HANDLING ROUTINE.
                      *        AFTER THE DUART GENERATES AN IRQ, DIRQ DETERMINES THE CAUSE OF
                      *        INTERRUPT. DIRQ CHECKS FOR THESE POSSIBLE CAUSES:
                      *
                      *                 COUNTER READY
                      *                 CHANGE IN CHANNEL A BREAK
                      *
                      *         ENTRY CONDITIONS:
                      *
                      *                 DUART'S INTERRUPT MASK HAS BEEN INITIALIZED.
                      *                 DUART HAS GENERATED AN INTERRUPT.
                      *
                      *         EXIT CONDITIONS:
                      *
                      *                 IF IRQ SOURCE IS:        THEN:
                      *                 ------------------       --------------------------------
                      *                 COUNTER                  SWAP TASKS BEING EXECUTED BY 68000
                      *                 CHANGE IN CH A BRK       EXIT TO MONITOR
                      *
                      *                 OTHERWISE, DIRQ RETURNS TO INTERRUPTED ROUTINE WITH
                      *                 ALL REGISTER CONTENTS UNALTERED.
                      *
                      *
                      *

000023F6 0839000300F0 DIRQ     BTST.B   #3,ISR                 WAS IRQ CAUSED BY THE COUNTER?
         000B
000023FE 6704                  BEQ      ABRKI                  NO, SKIP NEXT PART
00002400 6000FD24              BRA      SWPTSKS                YES, SWAP TASKS

00002404 0839000200F0 ABRKI    BTST.B   #2,ISR                 WAS IT A CHAN A BEGINNING-OF-BREAK IRQ?
         000B
0000240C 6736                  BEQ      DIRQR                  NO, SKIP NEXT PART
0000240E 13FC005000F0          MOVE.B   #$50,CRA               YES, CLEAR CHN A BRK IRQ BIT IN ISR
         0005
00002416 0839000200F0 ABRKI1   BTST.B   #2,ISR                 WAIT FOR END-OF-BREAK IRQ
         000B
0000241E 67F6                  BEQ      ABRKI1
00002420 13FC005000F0          MOVE.B   #$50,CRA               CLEAR CHN A BRK IRQ BIT IN ISR AGAIN
         0005
00002428 4A3900F00007          TST.B    RBA                    PULL BREAK CHARACTER FROM CHN A RX FIFO
```

```
0000242E 4BF82446               LEA.L    BRKMSG,A5               PRINT MESSAGE TO SCREEN
00002432 4DED0007               LEA.L    LBRKMSG(A5),A6
00002436 1E3C00F3               MOVE.B   #243,D7
0000243A 4E4E                   TRAP     #14
0000243C 2F7C00000000           MOVE.L   #MONITOR,2(A7)          NO, EXIT TO MONITOR
         0002

00002444 4E73          DIRQR    RTE

                       *
                       * MESSAGE STRINGS
                       *

00002446 000A          BRKMSG   DC.B     CR,LF                   BREAK RECEIVED MESSAGE
00002448 425245414B             DC.B     'BREAK'
         00000007      LBRKMSG  EQU      *-BRKMSG

0000244D 000A          CTRERR   DC.B     CR,LF                   COUNTER ERROR MESSAGE
0000244F 434F554E5445           DC.B     'COUNTER ERROR: IRQ NEVER RECEIVED'
         00000023      LCTRERR  EQU      *-CTRERR

00002470 000A          CHBMSG1  DC.B     CR,LF                   CHAN B TX NEVER READY MESSAGE
00002472 4348414E2042           DC.B     'CHAN B ERROR: TX NEVER READY TO TRANSMIT CHARACTER'
         00000034      LCHBMSG1 EQU      *-CHBMSG1

000024A4 000A          CHBMSG2  DC.B     CR,LF                   CHAN B RX NEVER READY MESSAGE
000024A6 4348414E2042           DC.B     'CHAN B ERROR: RX NEVER RECEIVED CHARACTER'
         0000002B      LCHBMSG2 EQU      *-CHBMSG2

000024CF 000A          CHBMSG3  DC.B     CR,LF                   CHAN B FRAMING ERROR MESSAGE
000024D1 4348414E2042           DC.B     'CHAN B ERROR: FRAMING ERROR'
         0000001D      LCHBMSG3 EQU      *-CHBMSG3

000024EC 000A          CHBMSG4  DC.B     CR,LF                   CHAN B PARITY ERROR MESSAGE
000024EE 4348414E2042           DC.B     'CHAN B ERROR: PARITY ERROR'
         0000001C      LCHBMSG4 EQU      *-CHBMSG4

00002508 000A          CHBMSG5  DC.B     CR,LF                   CHAN B INCORRECT CHAR REC'D MESSAGE
0000250A 4348414E2042           DC.B     'CHAN B ERROR: INCORRECT CHARACTER RECEIVED'
         0000002C      LCHBMSG5 EQU      *-CHBMSG5

00002534 000A          CHAMSG1  DC.B     CR,LF                   CHAN A TX NEVER READY MESSAGE
00002536 4348414E2041           DC.B     'CHAN A ERROR: TX NEVER READY TO TRANSMIT CHARACTER'
         00000034      LCHAMSG1 EQU      *-CHAMSG1

00002568 000A          CHAMSG2  DC.B     CR,LF                   CHAN A RX NEVER READY MESSAGE
0000256A 4348414E2041           DC.B     'CHAN A ERROR: RX NEVER RECEIVED CHARACTER'
         0000002B      LCHAMSG2 EQU      *-CHAMSG2

00002593 000A          CHAMSG3  DC.B     CR,LF                   CHAN A FRAMING ERROR MESSAGE
00002595 4348414E2041           DC.B     'CHAN A ERROR: FRAMING ERROR'
         0000001D      LCHAMSG3 EQU      *-CHAMSG3

000025B0 000A          CHAMSG4  DC.B     CR,LF                   CHAN A PARITY ERROR MESSAGE
000025B2 4348414E2041           DC.B     'CHAN A ERROR: PARITY ERROR'
         0000001C      LCHAMSG4 EQU      *-CHAMSG4

000025CC 000A          CHAMSG5  DC.B     CR,LF                   CHAN A INCORRECT CHAR REC'D MESSAGE
```

```
000025CE 4348414E2041          DC.B      'CHAN A ERROR: INCORRECT CHARACTER RECEIVED'
         0000002C     LCHAMSG5 EQU       *-CHAMSG5

                      *
                      * TEMPORARY STORAGE AREAS
                      *

         00007000              ORG       $7000

00007000 00000004     DTSKSSP  DS.L      1                    DORMANT TASK'S SYSTEM STACK POINTER

00007004 00000080     INBUF    DS.B      CSLNTH               INPUT BUFFER

00007084 00000001     PQIN     DS.B      1                    PRINT QUEUE INPUT POINTER
00007085 00000001     PQOUT    DS.B      1                    PRINT QUEUE OUTPUT POINTER
00007086 00000100     PQUE     DS.B      PQLNTH               PRINT QUEUE

00007186 00008000     PRTBUF   DS.B      PQLNTH*CSLNTH        PRINT BUFFER

                      *
                      * EXCEPTION VECTOR TABLE ENTRIES
                      *

         000003FC              ORG       $3FC

000003FC 000023F6     DIRQVEC  DC.L      DIRQ                 DIRQ EXCEPTION VECTOR

                               END

****** TOTAL ERRORS      0--
****** TOTAL WARNINGS    0--
```

SYMBOL TABLE LISTING

```
SYMBOL NAME      SECT   VALUE        SYMBOL NAME      SECT   VALUE

ABRKI                   00002404     INITTSK2                000020EA
ABRKI1                  00002416     INPTTSK                 0000211A
ACR                     00F00009     IP                      00F0001B
BRKMSG                  00002446     IPCR                    00F00009
BS                      00000008     IRQCNT                  0000FFFF
BSCHK                   000021E0     IRQMSK                  0000000C
BTRST                   00F0001F     ISR                     00F0000B
BTST                    00F0001D     ISSP                    00004000
CHAERR1                 00002078     ISTRG                   000021D2
CHAERR2                 00002088     IUSP                    00003800
CHAERR3                 00002098     IVR                     00F00019
CHAERR4                 000020A8     LBRKMSG                 00000007
CHAERR5                 000020B8     LCHAMSG1                00000034
CHAERRS                 00002074     LCHAMSG2                0000002B
CHAMSG1                 00002534     LCHAMSG3                0000001D
CHAMSG2                 00002568     LCHAMSG4                0000001C
CHAMSG3                 00002593     LCHAMSG5                0000002C
CHAMSG4                 000025B0     LCHBMSG1                00000034
CHAMSG5                 000025CC     LCHBMSG2                0000002B
CHANA                   00F00001     LCHBMSG3                0000001D
CHANB                   00F00011     LCHBMSG4                0000001C
CHASTS                  0000000C     LCHBMSG5                0000002C
CHBERR1                 00002020     LCTRERR                 00000023
CHBERR2                 00002032     LF                      0000000A
CHBERR3                 00002044     MONITOR                 00000000
CHBERR4                 00002054     MR1A                    00F00001
CHBERR5                 00002064     MR1B                    00F00011
CHBERRS                 0000201C     MR2A                    00F00001
CHBMSG1                 00002470     MR2B                    00F00011
CHBMSG2                 000024A4     OPCR                    00F0001B
CHBMSG3                 000024CF     OUTCH                   000023A2
CHBMSG4                 000024EC     OUTCH1                  000023B8
CHBMSG5                 00002508     OUTCHR                  000023CA
CHBSTS                  0000000E     POUTCH                  000023CC
CHCHK                   000022C4     POUTCH1                 000023E2
CHKA                    0000227A     POUTCHR                 000023F4
CHKB                    00002286     PQIN                    00007084
CHKCTR                  00002292     PQLMSK                  000000FF
CHRCHK                  00002324     PQLNTH                  00000100
CIRQ                    0000237A     PQOUT                   00007085
CIRQR                   0000238E     PQUE                    00007086
CLSB                    00F0000F     PRCHK                   00002318
CMSB                    00F0000D     PRNTTSK                 00002122
CR                      0000000D     PRTBUF                  00007186
CRA                     00F00005     PRTMSG                  00002000
CRB                     00F00015     PSSP                    00005000
CSLMSK                  0000007F     PSTRG                   00002204
CSLNTH                  00000080     PSTRG1                  0000220E
CSRA                    00F00003     PUSP                    00004800
CSRB                    00F00013     PUTCHAR                 000021EE
CTLR                    00F0000F     QSTRG                   0000214E
CTRCHK                  00002340     QSTRG1                  0000215E
```

```
DUART68S

CTRCHKR                 0000236A      QSTRG2                  0000217E
CTRERR                  0000244D      RBA                     00F00007
CTRERRS                 0000200C      RBB                     00F00017
CTRSTS                  00000010      RSTCHN                  00002332
CTUR                    00F0000D      RSTRG                   00002196
DINIT                   0000221E      RSTRG1                  000021A0
DINITR                  000022BE      RXCHK                   000022FA
DIRQ                    000023F6      RXCNT                   0000FFFF
DIRQR                   00002444      SNDCHR                  000022F0
DIRQVEC                 000003FC      SRA                     00F00003
DTSKSSP                 00007000      SRB                     00F00013
DUART                   00F00001      STPC                    00F0001F
ENABLA                  0000229A      STRC                    00F0001D
ENABLB                  000022B0      SWPTSKS                 00002126
ERRCHK                  000020C8      TBA                     00F00007
FRCHK                   0000230C      TBB                     00F00017
GETCHAR                 000021DC      TSKINIT                 00002000
IMR                     00F0000B      TXCHK                   000022DE
INBUF                   00007004      TXCNT                   0000FFFF
INCH                    00002390      WTIRQ                   00002360
INITTSK1                000020D8
```

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Employment Opportunity/ Affirmative Action Employer.

MOTOROLA *Semiconductor Products Inc.*

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

A17727-2 PRINTED IN USA 1-86 IMPERIAL LITHO C37103 10,000