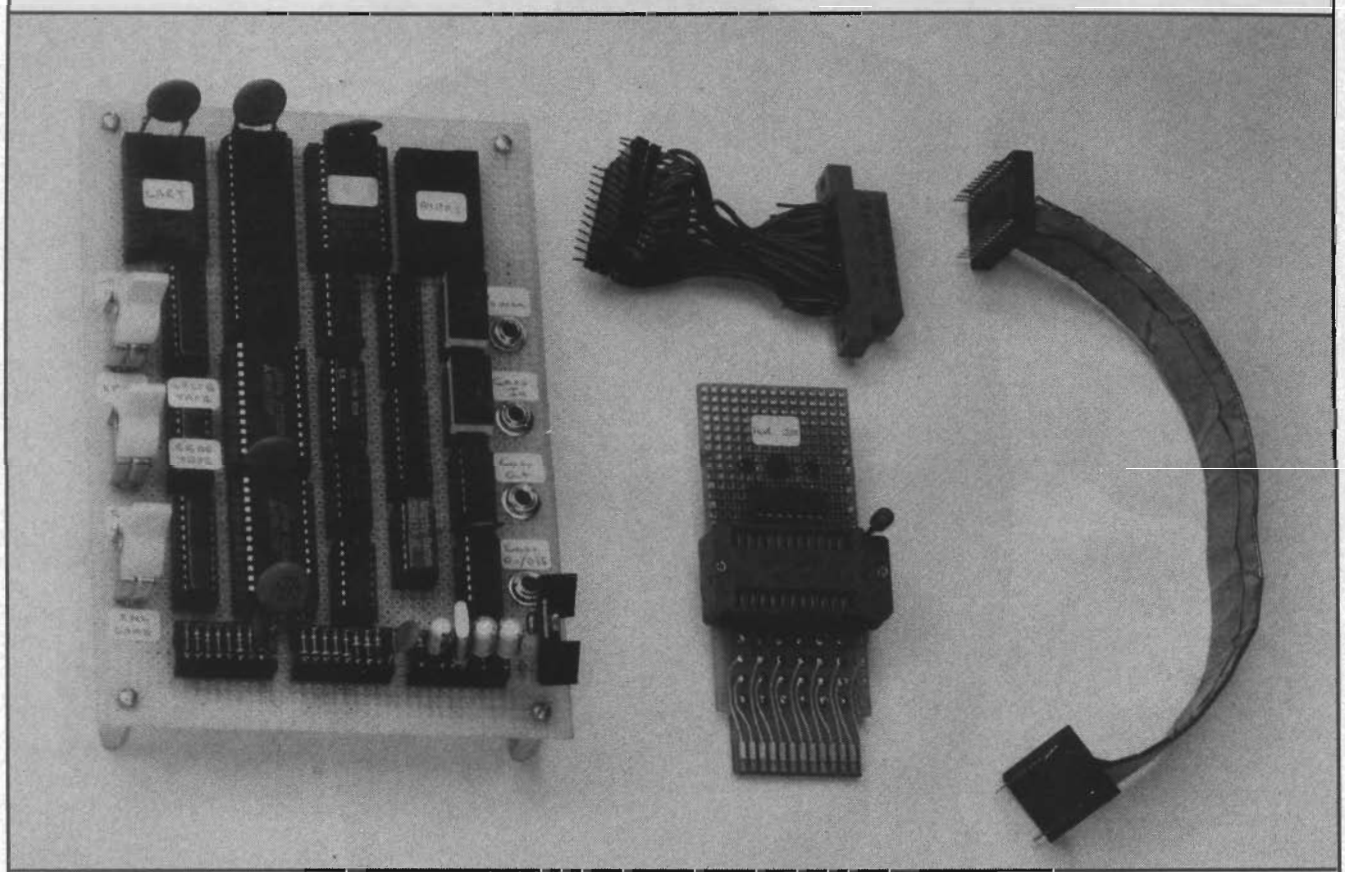
ATARI Game Recorder

GUY VACHON and DAVID A. CHAN



You can record the contents of your Atari 208 videogame cartridges on audio cassette tape! This month, we'll take a look at the software that's needed.

Part 2 LAST MONTH, WE looked at the basic approach we'll follow to store the contents of Atari *2600* game cartridges on audio cassette tape. We also looked at the hardware that's required, and briefly studied how cassette I/O is handled. This time, we'll look at the software in more depth. Then we'll see how we can build the game recorder and put it to use.

Game-recorder software

The complete software listing for the game recorder's operating system appears in Table 1. Note that it is written in Z80 mnemonics. Although we won't be discussing the software line by line, you might want to study Table 1 to get the details.

When we left off last time, we were discussing cassette I/O. Figure 5 showed a flowchart that described the cassette-read algorithm. Let's look at the software in more detail to see how it's used to detect the data and sync pulses. (Remember that sync pulses are sent out every 2 milliseconds. Data pulses are sent between the sync pulses—a pulse represents a 1 bit, while the lack of a pulse represents a zero bit.)

When the contents of a game cartridge is written to a cassette tape, a header of 2000 zero bits preceeds the actual beginning of the program bits. After the header, the game recorder also writes a (user-selected) label before each game. When the game recorder reads the contents of a cassette tape, its software looks for fifty consecutive zeros to decide that it has found the beginning of a game program. You might wonder why we write 2000 zero bits and look for only 50. There's a very practical reason: It allows the automatic gain control (AGC) of most recorders enough time to settle down.

After the recorder finds 50 consecutive zero bits, it keeps on looking until it finds a 1 bit. It then checks the name tag, which is output to the LAST GAME FOUND display. If the name tag matches the name of the game you selected, it keeps on reading bytes and storing them in the RAM. If the tag doesn't match, the game recorder keeps looking for another start-of-game header. (We'll give more details on that—and other operation aspects of the computer—a little later on in this article.)

You may recall that a parity bit is added to each instruction so that the game recorder will recognize when something

TABLE 1—GAME-RECORDER SOFTWARE

```
CURRENT_GAME_OUT            EQU  8000H
GAME_SEL_OUT                EQU  0A000H
INPUT                       EQU  0C000H
RAM_FIRST_BYTE_ADD          EQU  4000H
ROM_FIRST_BYTE_ADD          EQU  2000H
LAST_PLUS_1_BYTE_RAM_HIGH   EQU  50H
LAST_BYTE_ADD_RAM           EQU  4FFFH
;START OF INITIALIZATION
          .Z80
          LD   A,00H              ;CLEAR LEDS
          EXX
          LD   B,A
          LD   (CURRENT_GAME_OUT),A
          LD   A,3FH
          LD   C,A
          LD   (GAME_SEL_OUT),A
          EXX
;START OF MAIN PROGRAM
LOOP1:    LD   A,(INPUT)          ;SEE IF INC GAME
                                  ; PUSHED
          AND  01H
          JP   NZ, LAB1
          LD   A,0FFH             ;WAIT AND CHECK
                                  ; INPUT AGAIN
LOOPA:    DEC  A
          JP   NZ,LOOPA
          LD   A,(INPUT)
          JP   NZ,LAB1
          EXX                     ;START OF INC GAME
                                  ; PROGRAM
          LD   A,B
          INC  A
          AND  0FH
          LD   B,A
          EXX
          LD   IY,XX1
          JP   CONVERT
XX1:      EXX
          LD   C,A
          EXX
          LD   (GAME_SEL_OUT)A,
          LD   DE,7FFFH           ;WAIT HALF A SECOND
LOOPB:    DEC  A
          LD   A,E
          ADD  A,D
          JP   NZ,LOOPB
          JP   C,LOOPB

;CONTINUATION OF MAIN PROGRAM
LAB1:     LD   A,(INPUT)          ;SEE IF COPY PUSHED
          AND  02H
          JP   NZ,LAB2
          LD   A,0FFH             ;WAIT AND CHECK
                                  ; INPUT AGAIN
LOOPC:    DEC  A
          JP   NZ,LOOPC
          LD   A,(INPUT)
          AND  02H
          JP   NZ,LAB2
          LD   DE,RAM_FIRST_BYTE_ADD
                                  ;START OF COPY
                                  ; PROGRAM
          LD   HL,ROM_FIRST_BYTE_ADD
          LD   BC.1000H
          LDIR

;CONTINUATION OF MAIN PROGRAM
LAB2:     LD   A,(INPUT)          ;SEE IF DOWNLOAD
                                  ; PUSHED
          AND  04H
          JP   NZ,LAB3
          LD   A,0FFH             ;WAIT AND CHECK
                                  ; INPUT AGAIN
LOOPD:    DEC  A
          JP   NZ,LOOPD
          LD   A,(INPUT)
          AND  04H
          JP   NZ,LAB3
          EXX                     ;START OF DOWNLOAD
                                  ; PROGRAM
          LD   A,C                ;TURN ON DECIMAL
                                  ; POINT
          EXX
          OR   80H
          LD   (GAME_SEL_OUT),A
BLOCIN:   LD   DE,01F4H
LOOPE:    LD   HL,XX2             ;FIND 500 ZEROS
          JP   BITIN
XX2:      JP   C,BLOCIN
          DEC  DE
          LD   A,E
          ADD  A,D
          JP   NZ,LOOPE
          JP   C,LOOPE
LOOPF:    LD   LH,XX3             ;FIND 1ST BIT OF 1ST
                                  ; BYTE
          JP   BITIN
XX3;      HP   NC,LOOPF
          LD   A,01H              ;GET 1ST BYTE
          LD   B,07H
          LD   IX,XX4
          JP   BYTEIN
XX4;      AND  0FH                ;CONVERT TO 7 SEG
                                  ; AND DISPLAY
          LD   IY,XX5
          JP   CONVERT
XX5:      LD   (CURRENT_GAME_OUT),A
          LD   D,A                ;SEE IF CORRECT
                                  ; GAME
          EXX
          LD   A,C
          EXX
          CP   D
          JP   NZ,BLOCIN
          JD   DE,RAM_FIRST_BYTE_ADD
                                  ;GET REST OF BLOCK
LOOPG:    LD   A,00H
          LD   B,08H
          LD   IX,XX6
          JP   BYTEIN
XX6:      LD   (DE),A
          INC  DE
          LD   A,D
          CP   LAST_PLUS_1_BYTE_RAM_HIGH
          JP   NZ,LOOPG
          LD   DE,0FFFFH          ;WAIT ONE SECOND
LOOPH:    DEC  DE
          LD   A,E
          ADD  A,D
          JP   NZ,LOOPH
          JP   C,LOOPH
          EXX                     ;TURN OFF DECIMAL
                                  ; POINT
          LD   A,C
          EXX
          LD   (GAME_SEL_OUT),A

;CONTINUATION OF MAIN PROGRAM
LAB3:     LD   A,(INPUT)          ;SEE IF RECORD
                                  ; PUSHED
          AND  08H
          JP   NZ,LAB4
          LD   A,0FFH             ;WAIT AND CHECK
                                  ; AGAIN
LOOPI;    DEC  A
```

TABLE 1 (continued)

```
        JP    NZ,LOOP1
        LD    A,(INPUT)
        AND   08H
        JP    NZ,LAB4
        EXX                       ;START OF RECORD
                                  ; PROGRAM
        LD    A,C                 ;TURN ON DECIMAL
                                  ; POINT
        EXX
        OR    80H
        LD    (GAME_SEL_OUT),A
        LD    DE,07D0H
LOOPJ:  AND   0FFH                ;OUTPUT 2000 ZEROS
        LD    HL,XX7
        JP    BITOUT
XX7:    DEC   DE
        LD    A,E
        ADD   A,D
        JP    NZ,LOOPJ
        JP    C,LOOPJ
        EXX                       ;OUTPUT BLOCK
                                  ; ADDRESS
        LD    A,B
        EXX
        OR    0F0H
        LD    IX,XX8
        JP    BYTEOUT
XX8:    LD    DE,RAM_FIRST_BYTE_ADD
                                  ;OUTPUT BLOCK
LOOPK:  LD    A,(DE)
        LD    IX,XX9
        JP    BYTEOUT
XX9:    INC   DE
        CP    LAST_PLUS_1_BYTE_RAM_HIGH
        JP    NZ,LOOPK
        LD    DE,0FFFFH           ;WAIT ONE SECOND
LOOPL:  DEC   DE
        LD    A,E
        ADD   A,D
        JP    NZ,LOOPL
        JP    C,LOOPL
        EXX                       ;TURN OFF DECIMAL
                                  ; POINT
        LD    A,C
        EXX
        LD    (GAME_SEL_OUT),A
LAB4:   JP    LOOP1

;START OF SUBROUTINES
;BYTEIN SUBROUTINE - GETS ONE BYTE FROM TAPE
;IX = RETURN ADDRESS
;GIVEN: A IS (EMPTY) BYTE
;       B IS # BITS LEFT
;RESULT: A IS BYTE
;USES: B
;CALLS: BITIN
;CANNOT AFFECT: DE
BYTEIN: LD    HL,XX10             ;START OF BYTEIN
                                  ; PROGRAM
        JP    BITIN               ;GET ENTIRE BYTE
XX10:   RLA
        DEC   B
        JP    NZ,BYTEIN
        AND   0FFH                ;COMPUTE PARITY
        EX    AF,AF'
        LD    HL,XX11             ;GET PARITY BIT
        JP    BITIN
XX11:   JP    C,LAB5              ;SEE IF ERROR BY
                                  ; CHECKING 'CARRY'
                                  ; THEN
        EX    AF,AF'
        JP    PE,PERROR
        JP    (IX)
LAB5:   EX    AF,AF'              ;RECALL BYTE &
                                  ; FLAGS AND CHECK
                                  ; PARITY
        JP    PO,PERROR
        JP    (IX)
PERROR: LD    A,94H               ;ERROR, DISPLAY
                                  ; MESSAGE
        LD    (CURRENT_GAME_OUT),A
        LD    DE,LAST_BYTE_ADD_RAM
        JP    (IX)

;BITIN SUBROUTINE - GETS ONE BIT FROM TAPE
;HL = RETURN ADDRESS
;RESULT: BIT IS CARRY
;USES: D',E',H'
;CANNOT AFFECT: DE,B,A

BITIN:  EXX                       ;START OF BITIN
                                  ; PROGRAM
        LD    D,A                 ;EXCHANGE
                                  ; REGISTERS AND
                                  ; STORE A
LOOPM:  LD    A,(INPUT)           ;FIND HIGH
        AND   10H
        JP    Z,LOOPM
        LD    E,8FH               ;WAIT 1MSEC
LOOPN:  DEC   E
        JP    NZ,LOOPN
        LD    E,0CH               ;SEE IF 1 OR 0 FOR
                                  ; 0.25MSEC
LOOPO:  LD    A,(INPUT)
        AND   10H
        CCF
        JP    NZ,LAB6
        DEC   E
        JP    NZ,LOOPO
        CCF
LAB6:   LD    E,5DH               ;WAIT 0.65MSEC
LOOPP:  DEC   E
        JP    NZ,LOOPP
        LD    A,D                 ;RECALL A,
                                  ; EXCHANGE REGS &
                                  ; RETURN
        EXX
        JP    (HL)

;BYTEOUT SUBROUTINE - WRITES A BYTE ONTO TAPE
;IX = RETURN ADDRESS
;GIVEN: A IS BYTE
;USES: B
;CALLS: BITOUT
;CANNOT AFFECT:DE
BYTEOUT: LD   B,08H               ;START OF BYTEOUT
                                  ; PROGRAM
LOOPQ:  RLCA                      ;OUTPUT BYTE
        LD    HL,XX12
        JP    BITOUT
XX12:   DEC   B
        JP    NZ,LOOPQ
        AND   0FFH                ;COMPUTE AND
                                  ; OUTPUT PARITY
        JP    PO,LAB7
        CCF
LAB7:   LD    HL,XX13
        JP    BITOUT
XX13:   JP    (IX)

;BITOUT SUBROUTINE - WRITES A BIT ONTO TAPE
;HL = RETURN ADDRESS
;GIVEN: CARRY IS BIT
;USES: C
;CALLS: PULSE
;CANNOT AFFECT:DE,B,A
```

TABLE 1 (continued)

```
BITOUT:  EX   AF,AF'          ;START OF BITOUT
                              ; PROGRAM
         SCF                  ;STORE ORIGINAL
                              ; BYTE AND FLAGS
         LD   IY,XX14         ;OUTPUT 1ST PULSE
         JP   PULSE
XX14:    LD   C,6BH           ;WAIT
LOOPR:   DEC  C
         JP   NZ,LOOPR
         EX   AF,AF'          ;GET ORIGINAL FLAGS
                              ; AND BYTE
         LD   C,A             ;STORE ORIGINAL
                              ; BYTE ONLY
         EX   AF,AF'
         LD   A,C
         EX   AF,AF'
         LD   IY,XX15         ;OUTPUT 2ND PULSE
         JP   PULSE
XX15:    LD   C,6BH           ;WAIT
LOOPS:   DEC  C
         JP   NZ,LOOPS
         EX   AF,AF'          ;RECALL ORIGINAL
                              ; BYTE AND RETURN
         JP   (HL)

;SUBROUTINE PULSE - WRITE A PULSE ONTO TAPE
;IY = RETURN ADDRESS
;GIVEN: PULSE IF CARRY
;USES: C
;CANNOT AFFECT:DE,B

PULSE:   LD   A,ØØH           ;START OF PULSE
                              ; PROGRAM
         JP   NC,LAB8         ;SET OUT IF
                              ; REQUIRED
         OR   8ØH
LAB8:    LD   (CURRENT_GAME_OUT),A
                              ;OUT AND WAIT
         LD   C,24H
LOOPT:   DEC  C
         JP   NZ,LOOPT
         LD   A,ØØH           ;TURN OFF
         LD   (CURRENT_GAME_OUT),A
         JP   (IY)

;SUBROUTINE CONVERT - CONVERTS DATA TO 7 SEGMENT
;IY = RETURN ADDRESS
;GIVEN: A IS TO BE CONVERTED
;RESULT: A IS CONVERTED DATA

CONVERT: CP   ØØH             ;START OF CONVERT
                              PROGRAM
         JP   NZ,LABA
         LD   A,3FH
         JP   (IY)
LABA:    CP   Ø1H
         JP   NZ,LABB
LABB:    CP   Ø2H
         JP   NZ,LABC
         LD   A,5BH
         JP   (IY)
LABC:    CP   Ø3H
         JP   NZ,LABD
         LD   A,4FH
         JP   (IY)
LABD:    CP   Ø4H
         JP   NZ,LABE
         LD   A,66H
         JP   (IY)
LABE:    CP   Ø5H
         JP   NZ,LABF
         LD   A,6DH
         JP   (IY)
LABF:    CP   Ø6H
         JP   NZ,LABG
         LD   A,7DH
         JP   (IY)
LABG:    CP   Ø7H
         JP   NZ,LABH
         LD   A,Ø7H
         JP   (IY)
LABH:    CP   Ø8H
         JP   NZ,LABI
         LD   A,7FH
         JP   (IY)
LABI:    CP   Ø9H
         JP   NZ,LABJ
         LD   A,6FH
         JP   (IY)
LABJ:    CP   ØAH
         JP   NZ,LABK
         LD   A,77H
         JP   (IY)
LABK:    CP   ØBH
         JP   NZ,LABL
         LD   A,7CH
         JP   (IY)
LABL:    CP   ØCH
         JP   NZ,LABM
         LD   A,39H
         JP   (IY)
LABM:    CP   ØDH
         JP   NZ,LABN
         LD   A,5EH
         JP   (IY)
LABN:    CP   ØEH
         JP   NZ,LABO
         LD   A,79H
         JP   (IY)
LABO:    LD   A,71H
         JP   (IY)
         END
```

has been misrecorded. If incorrect parity is detected when the computer is reading from the tape, it will stop reading, and the LAST GAME FOUND display will show a message of three horizontal bars to indicate an error.

Before we go any further, we should talk a little about the memory mapping used in the game recorder. The system ROM resides from ØØØØH to 1FFFH. (Note that a capital "H" indicates that a number is written in hexadecimal.) The game cartridge occupies the second 8K block—2ØØØH to 3FFFH. The game recorder's RAM is located from 4ØØØ to 5FFFH. Cassette I/O and the displays are also memory mapped: The block from 8ØØ ØH to 9FFFH is used for the LAST GAME FOUND display and the cassette data output, while the block from AØØØH to BFFFH is used for the GAME SELECTED display and for the remote cassette control. The cassette data input and the switches are memory mapped from CØØ ØH to DFFFH. Note that two 8K blocks (6 ØØØH–7FFFH and EØØØH–FFFFH) are not used.

The easiest job that our computer has to do is to read the program ROM. As it operates now, the computer can copy all 2K × 8 ROM's and 4K × 8 ROM's. As you might expect, it is possible to modify the recorder to copy 8K × 8 ROM's. Note, for example, that although an 8K block was left available for program-storage RAM, the hardware as presented has provision for only 4K.

We'll talk more about how to expand the unit to record larger programs, and show you how to build and use it, when we continue our look at the Atari game recorder next time. **R-E**