PCROM.Overview 3:19pm Tuesday, 2 August 1983

The TI 99/4A home computer requires cartridges with a GROM, a patented device that TI is selling. The code on the GROM is an interpreted machine language (much like Pascal P-code) called GPL. We are making a PCROM, that looks like a GROM, without violating TI's patent.

We have been developing game code in 9900 assembly language (the CPU on the 99/4). In order to produce a cartridge, we have written code for the PCROM that will begin execution of the 9900 game code on another ROM. This code uses less than 512 bytes of the 6K available on the PCROM.

## Game Specific Storage

Due to GPL's slow excution and use of precious RAM, the use of GPL subroutines is not attractive during game play. However, 5.5K of the PCROM can be used for game data storage. The PCROM is read in a fashion similar to the way that the VRAM is written to through the 9918. After 2 bytes of address data is written to the PCROM, data can be read from it. The PCROM's internal address pointer is auto-incremented between reads. Graphics and sound data are good candidates for tables to be put on PCROM, and are more easily frozen 6 weeks earlier than code, this being required by the PCROM lead time.

The mechanism to call GPL subroutines from 9900 code has been identified and documentated. There are GPL subroutines in the 99/4A's operating system. A GPL assembler also exists to allow writing of GPL code for the PCROM. The problem with any effort to use GPL for games is the price of speed and RAM usage.

The PCROM allows startup of 9900 code with the option of asking the gamester for the number of players and passing that number to the 9900 program. The bulk of the storage on the PCROM is available for game specific data. There is also the capability for the PCROM to generate a single output line for bank switching.

## Sound for TI99/4A

The OS on the 99/4A has a workable sound routine which requires rather large tables containing much redundant data. P B does have a routine which converts sound tables produced from our sound lab, into the tables required by TI's sound routine. The conversion routine is about 200 bytes of 9900 code, and sound tables for games average about 800 to 1000 bytes.

Now the detail stuff.
4:41pm  Monday, 1 August 1983

## PCROM SPECIFICATION

A PCROM, Program Counter Read Only Memory, is a sequential storage unit which has an on-chip auto-incrementing address counter. It is designed to contain Graphic Programming Language Code, which is executed by a GPL interpreter in the TI99/4A personal computer.

*6144x8 read only memory

*16 pin package

*Single power supply 4.75 to 5.25 volts

It is a memory mapped device which functions like the VDP RAM. The memory is addressed by writing it's address to a specific CPU address and reading data from another specific address. PCROM addresses are from 0000 through F7FF hex. Each one has 6K bytes of memory that start from an even-numbered first digit address. For example, PCROM 0 is at address 0000 through 17FF and PCROM 1 is at address 2000 throuh 37FF. The computer can access up to eight PCROMs at a time.

## Circuit Description

The PCROM interfaces to the CPU through a parallel 8-bit bidirectional data bus, chip enable line, ready line, two select lines,OS and R/W, and a clock input. Both the data and address information is bused through the eight parallel data lines with the proper mode select address determining the functional operation.

## I/O Operations

When CE\ becomes active low the mode select lines determine one of the four I/O operations.

INCREMENT ADDRESS:
The address counter is incremented
without performing a read data operation.

READ DATA:
The read data operation transfers the data byte
to the data register pointed to by the address counter.
The data is then transfered to the cpu if the device
is in the current page.The address counter is then
auto-incremented.
Note: Upon power-up of the PCROM a read data operation
is required to initialize the device.

WRITE ADDRESS:

The write address operation transfers the data byte on the data I/O bus to the least significant byte of the address counter and the previous byte in the least significant byte of the address counter is transfered to the most significant byte.Two consecutive write addresses cause a complete address to be transferred to the address counter.

READ ADDRESS:

The read address operation transfers the most significant byte of the address counter on to the data bus and the least significant byte of the counter is transfered to the most significant byte of the counter. Two consecutive read operations are required to read the complete address and new address should be written at this time.

## PCROM USE WITH THE TI 99/4A

Menu Items:

After the TI Logo screen, the menu screen will list "PARKER BROTHERS' GAME" after "TI BASIC"

Startup:

On selection of "PARKER BROTHERS' GAME", execution will begin in GPL on the PCROM. If the PCROM code finds a >55 at address >6000 (where the 9900 ROM resides), the GPL code will check address >6001 and if non-zero,the gamester is asked for number of players. The low order 4 bits of address >6001 defines the maximum number of players allowed (values greater than 9 and the high order 4 bits are reserved for future use). The number of players is then passed to the 9900 program in R0. If data at >6000 is not >55 GPL does a SYTEM RESET. Due to the way code start up works (see below) the users WP in the BLWP vector @>6002 can not = >830E or >8310.

```
PSEUDO CODE (WHAT GPL IS DOING):
       IF (>6001) = >55yx
        THEN DO
         IF x > 0
          THEN DO
           ASK FOR # OF PLAYERS
           WHERE x IS MAX VALID ENTRY
          END
         LOAD FOLLOWING 9900 CODE @>8300-8312
         ***************************************************
         *                                                 *
         *ADDR DATA                                        *
         *---- ----                                        *
         *8300 8302    DATA             ;GPL USES TO START *
         *8302 02E0    LWPI >8300       ;USE RAM AT >8300  *
         *8304 8300                                        *
         *8306 C020    MOV  @>6002,R0   ;PNT TO USER WP    *
         *8308 6002                                        *
         *830A D420    MOVB @>8312,*R0  ;STORE # PLAYERS   *
         *                              ;IN USERS R0        *
         *830C 8312                                        *
         *830E 0420    BLWP @>6002      ;START USERS CODE  *
         *8310 6002                                        *
         *8312 nnzz    DATA             ;GPL PLACES # OF   *
         *                              ;PLAYERS HERE       *
         ***************************************************

         JMP TO 9900 CODE @>8302 (JUST LOADED)
        ELSE DO
         SYSTEM RESET

START UP VECTOR EXAMPLE:

         ***************************************************
         *                                                 *
         *ADDR DATA                                        *
         *---- ----                                        *
         *6000 55yx    DATA             ;x IS MAX # OF     *
         *                              ;VALID PLAYERS      *
         *6002 USERWP  DATA             ;VECTOR USED BY    *
         *6004 USERPC  DATA             ;BLWP TO START     *
         *                              ;USER PROGRAM       *
         ***************************************************
```

Sample 9900 Code for accessing the PCROM

```
*****************************************************************
* THIS PROGRAM READS DATA FROM THE PCROM UNTIL THE BYTE READ IS 0
* THIS DATA COULD BE SOME KIND TABLE WITH A VALUE OF 0 FOR THE TERMINATOR
* >>>> NOTE ANY ADDRESSING MODE MAY BE USED TO ACCESS THE PCROM
*
*
PCRMRD  EQU     >9800           PCROM READ DATA ADDR
PCRMWA  EQU     >9C02           PCROM WRITE ADDR REGISTER
PCRMRA  EQU     >9802           PCROM READ ADDR REGISTER
*
*
*
        LI      R14,TBLSTRT     LOAD R14 WITH PCROM TABLE STARTING ADDR
*
*
*
*                               LOOP TO LOAD DATA FROM PCROM
LDADDR
        BL      @INITAD         R14 CONTAINS PCROM TABLE ADDR
LDLOOP
        MOVB    @PCRMRD,*R0+    READ DATA FROM PCROM & STORE IT
*                               IF AT END OF TABLE (DATA READ = 0)
        JEQ     TBLEND           THEN GO PROCESS TABLE
        JMP     LDLOOOP          ELSE GO READ DATA FROM PCROM
*
*
*
*
* >>>> NOTE: THE PCROM ADDR DOES NOT HAVE TO BE SAVED BETWEEN ACCESSES
* >>>>       IF THE PCROM ADDR IS NOT WRITTEN TO.
*
*
* >>>> NOTE: READING THE PCROM ADDR DESTROYS THE VALUE IN THE ADDR REGISTER
*
*
*    THIS IS AN EXAMPLE OF HOW TO READ THE PCROM ADDR
*    THE PCROM ADDR WILL HAVE TO BE RE-INITIALIZED BEFORE ANY PCROM READS
*
TBLEND
*                               SAVE PCROM CURRENT ADDR IN R14
        MOVB    @PCRMRA,R14     GET MSBYTE OF PCROM ADDR
        SWPB    R14             SAVE MSB OF ADDR IN LSB OF R14
        MOVB    @PCRMRA,R14     GET LSBYTE OF PCROM ADDR
        SWPB    R14             SWITCH SO THAT MSB OF R14 CONTAINS MSB OF PCROM
*
*
*
*    THIS IS WHERE THE CODE WOULD BE TO PROCESS THE TABLE JUST READ IN
*
*
*
        JMP     LDADDR          LOOP TO READ IN ANOTHER TABLE
*
*
*
```

```
*    THIS IS AN EXAMPLE OF A SUBROUTINE TO INITIALIZE THE PCROM ADDR REGISTER
*
INITAD
*                                   SET PCROM ADDR TO VALUE CONTAINED IN R14
       MOVB    R14,@PCRMWA          WRITE MSB OF ADDR TO PCROM FIRST
*
*  >>>> NOTE: A DELAY IS REQUIRED BETWEEN ADDR WRITES <<<<
*
       SWPB    R14                  GET LSB
       MOVB    R14,@PCRMWA          WRITE LSB OF PCROM ADDR
       RT
*********************************************************************************
```

PCROM Bank Switching
---

In order to select different banks of 9900 rom, it is necessary to have the appropriate PCROM address in the PCROM address register. The 13th least significant bit of the PCROM address corresponds to the 13th least significant bit of the 9900 address. The PCROM address written determines which bank is selected. The following section of code is the recommended method to call subroutines in the other bank.

```
MSK1     EQU  >1FFF          USED TO STIRP MSB(ITS) OF ADDR & SAVE LSB(ITS)
CURBNK   EQU  >y000          y IS THE CURRENT BANK FROM WHICH THE SUBROUTINE
*                            CALL IS BEING MADE y = 0,2,4, OR 6 (0 & 4 AND
*                            2 & 6) THE 2ND LSB OF y IS A DON'T CARE
PCRMWA   EQU  >9C02          PCROM WRITE ADDR REGISTER
*
*
*
* FORM "PSEUDO ADDRESS" BY GETTING LEAEST SIGNIFICANT BITS OF ADDRESS...
* OFFSET INTO PAGE AND THEN ADDING THE BANK ADDRESS TO THE MSB(ITS)
*
CALSUB   LI   R10,SUBTST     GET SUBROUTINE ADDRESS
         ANDI R10,MSK1       STRIP OFF MSB(ITS)
         AI   R10,x000       SELECT WHICH BANK x = 0,2,4, OR 6 (0 & 4 AND
*                            2 & 6) THE 2ND LSB OF x IS A DON'T CARE
         BL   @BANKSW        GO SWITCH BANKS & CALL SUBROUTINE
*
*
*
BANKSW   MOV  R11,R9         SAVE RETURN ADDR
         ANDI R9,MSK1        CONVRT TO PSEUDO ADDR
         ORI  R9,CURBNK      CURBNK IS THE CURRENT BANK CODE
SWITCH   MOVB R10,@PCRMWA    SET BANK BY LOADING
         MOVB R10,@PCRMWA    LOAD DUMY LOW ORDER PART
         ORI  R10,>6000      MAKE PSEUDO ADDR REAL
         BL   *R10           CALL THE REQUESTED SUBR
         MOV  R9,R10         GET RETURN PSEUDO ADDR
         JMP  SWITCH         AND GO THERE
*
*
*
* >>>> NOTE: THIS ROUTINE NEEDS R9,R10,R11 PRESERVED <<<<
```

In the above routine the bank switch will occur after the 2nd byte of PCROM address is written. A copy of this routine must reside in each bank at the same address relative to the start of each bank

==========

I won't be here on Friday, so if you have any questions, tuff.
========