PCMCIA RELEASE 2.0 INTERFACE BOARD FOR DRAGONBALL UPDATE

DATE : 24 NOV 98

DTACK GENERATOR

The DTACK Generator is a state machine. It delays the memory or I/O access cycle of the PC Card when the card asserts the *WAIT signal.

The state diagram is shown in Figure 0-1

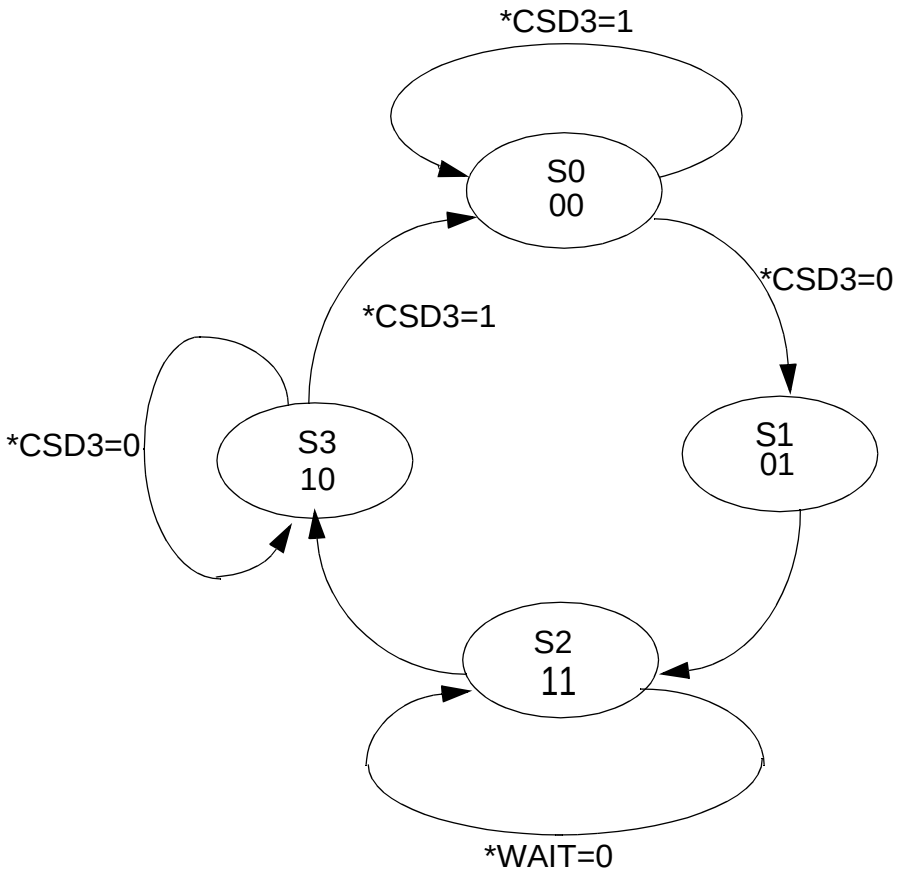
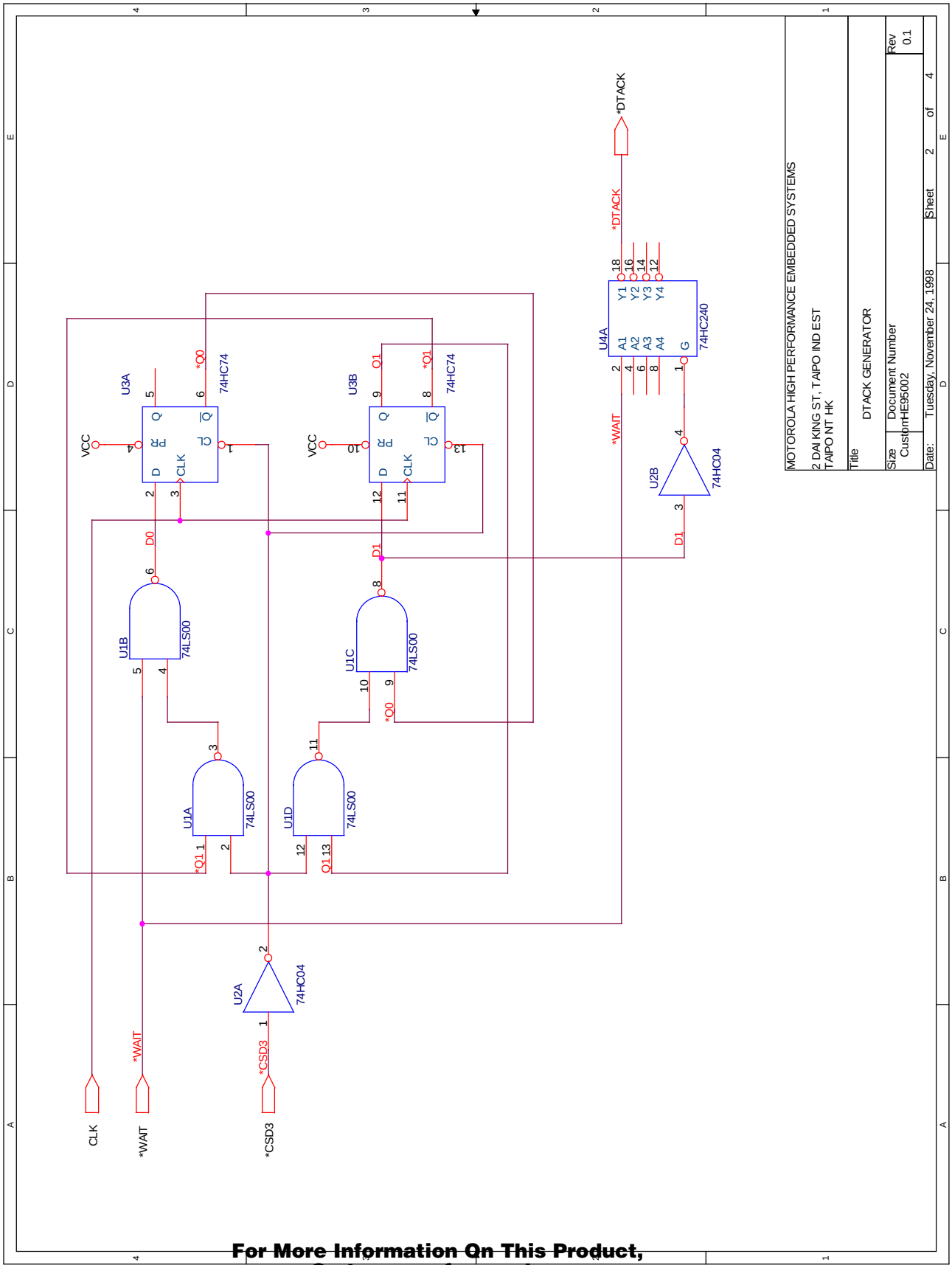


Figure 0-1. DTACK Generator State Diagram

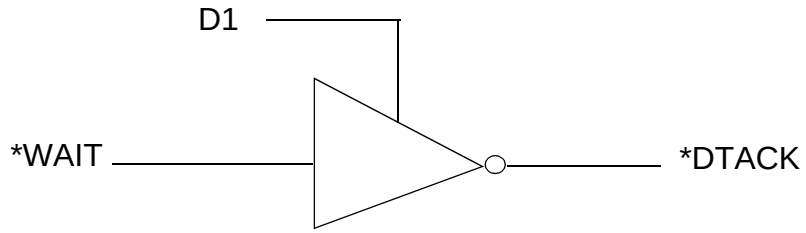
All the state transitions take place at rising edge of the system clock.

The state machine can be implemented by following circuit.

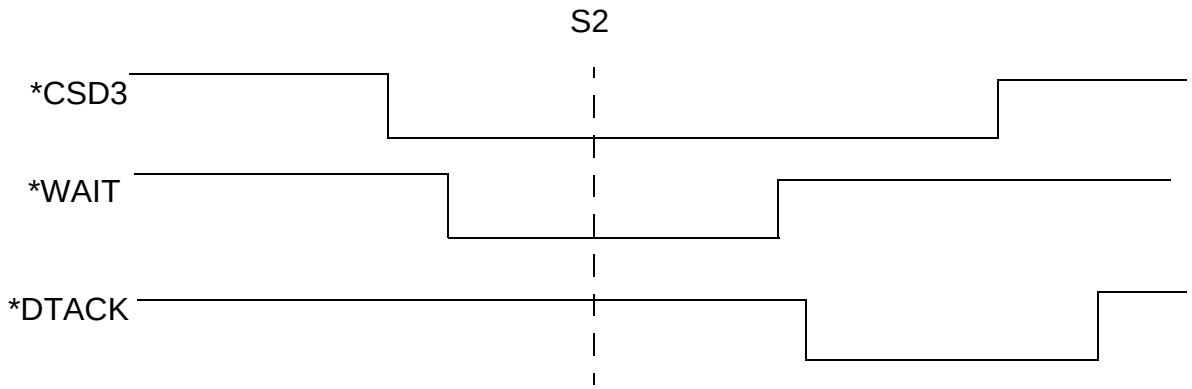


State S0 stands for the time when both *CSD3 is negated. There is no card access activities.

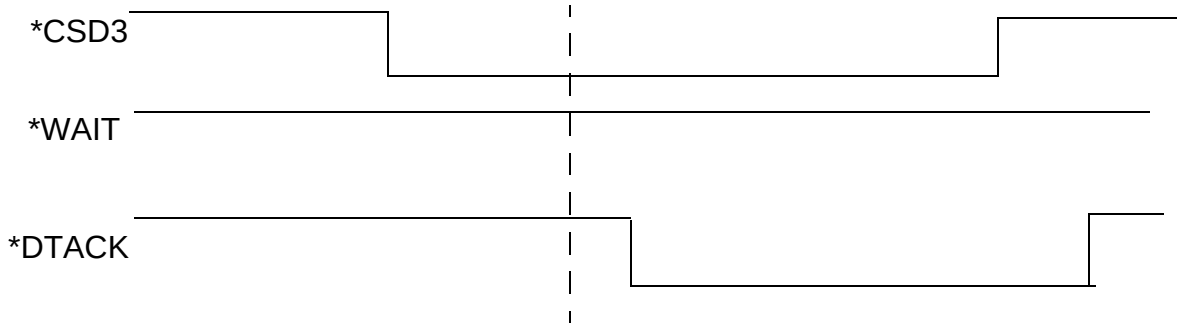
When *CSD3 is asserted, the state machine transits to state S1. Then after a clock , it transits to state S2. After the PC Card has asserted the *WAIT signal, the state machine holds in state S2. Beginning from S2, *DTACK is generated from inverted *WAIT signal which is implemented as follows.



When state machine is in S2 or S3, D1 asserts and enables the inverter. Therefore when *WAIT asserts, *DTACK is negated and when *WAIT negates, *DTACK is asserted.



When there is no *WAIT asserting in the cycle, *DTACK will assert after S2 until S0.



After *WAIT signal is negated, the state machine transits to S3. When the *CSD3 negates, it returns to S0 and D1 negates and disables the invertor so that *DTACK returns to high impedance state.

DECODER

It is implemented by a PAL22V10. It decodes the DragonBall control signals and generates the PCMCIA control signals.

In this application, all the I/O space and memory space incluсding attribute memory and common memory in the PC card are selected by *CSD3. The DragonBall has to assert *REG(PM2 = 0), while accessing the attribute memory space. To access common memory, DragonBall has to negate *REG(PM2=1). To access I/O, DragonBall has to assert IOEN (PJ4) and *REG.

The following is the updated PAL equation with only one chip select used.

```
TITLE          PCMCIA VER 2.01 DECODER
AUTHOR         JACKY LI
COMPANY        MOTOROLA INC.
DATE           23/11/98
REVISION
PATTERN

CHIP            U1      PAL22V10
;PIN    1    2     3     4     5     6     7     8     9     10   11     12
        CLK  IOEN /CSD3 /UDS  /LDS  /LWE  NC   /OE   /WE   /UWE  /BUFF  GND

;PIN   13  14   15    16   17    18    19    20    21   22     23    24
      NC  Q    DIR   /G   /BOE  /BWE  /IOW  /IOR  NC   /CE2   /CE1  VCC GLOBAL

EQUATIONS

CE1 = UDS * CSD3 + LDS * CSD3
CE1.TRST = BUFF

CE2 = UDS * LDS * CSD3
CE2.TRST = BUFF

IOR = CSD3 * IOEN * OE
IOR.TRST = BUFF

IOW = CSD3 * IOEN * LWE + CSD3 * IOEN * UWE
IOW.TRST = BUFF

BWE = /IOEN * WE * CSD3
BWE.TRST = BUFF

G = CSD3 * BUFF

DIR = /OE + WE

BOE = /IOEN * OE * CSD3
BOE.TRST = BUFF
```

As you can see from the PAL equation, the signals are generated by simple logic, therefore logic gates can be used instead of PAL to implement this decoder.

0.1 MEMORY MAP

The PCMCIA interface memory map is shown in Fig 0-2.

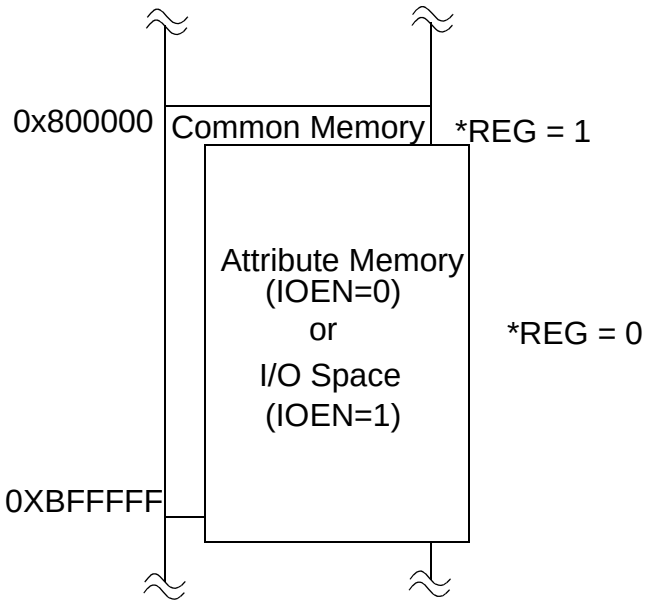


Figure 0-2. PCMCIA Interface Memory Map

0.2 INITIALIZATION CODE

```
In Init_CS, PJ4 is set as I/O port. (write 0xfff43b=0x1F)
CSD3 selects address range 0x800000 to 0xBFFFFF
and external DTACK is used.


nop *************BEFORE INSERTING CARD ************
nop Init_CS
write 0xfff43b -b = 0x1F
write 0xfff14c -l = 0x80013ff7

nop Buffer_Off
write 0xfff441 -b = 0xFE

nop AttrMem_Off
write 0xfff44b -b = 0x06
write 0xfff449 -b = 0xff
write 0xfff448 -b = 0x00


nop *************AFTER INSERTING CARD ************
```

```
nop Vcc_On
write 0xfff44b -b = 0x1E
write 0xfff449 -b = 0xE7
write 0xfff448 -b = 0x1C

nop Buffer_On
write 0xfff443 -b = 0x3C
write 0xfff441 -b = 0xEE
write 0xfff440 -b = 0x10

nop C_Reset
write 0xfff44b -b = 0x1E
write 0xfff448 -b = 0x1C
write 0xfff449 -b = 0xE7

nop AttrMem_On
write 0xfff44b -b = 0x1E
write 0xfff449 -b = 0xE3
write 0xfff448 -b = 0x1C

**********************************

To read attribute memory,
type
read 0x8000000

To read I/O space,
type
write 0x810000 -b = 0x41
write 0xfff438 -w = 0x10f5

Note: Port PJ4 is set to output port and to assert IOEN.
```