다중 컴퓨터와 다중 터미날을 상호 접속 하기위한 T(time)-스위칭 시스템의 구현

나 종 래, 김 경 진, 변 증 남, 정 명 진
한국 과학 기술원 전기및 전자 공학과

An Implementation of T(time)-switching system for the interconnection of multiple terminals and computers

Jong-Ray Na, Kyung-Jin Kim, Zeung-Nam Bien, Myung-Jin Chung
Department of electical and electronics, KAIST

Abstract

A time-division-multiplexed switching system is implemented for connecting various computers and terminals in the laboratory environment.

1. 서 론

컴퓨터실에 여러 기종의 컴퓨터 라인이 연결되어 있고 또한 다수의 터미날이 있는 경우 사용자마다 기종이 다를 수 있고 각 기종에 대한 사용 빈도가 다르게 된다. 이 경우 모든 컴퓨터에 각각의 전용 터미날을 두기보다는 각 컴퓨터의 RS-232 connector를 빈번히 기계적으로 교체해가며 사용하게 된다. 또한 멀리 떨어져 있는 하나의 터미날로 여러 기종의 컴퓨터에 접속하고자 할 경우 computer network에 의한 해결 방법이 있다. 이 경우 프로토콜의 실현을 위해 많은 비용과 시간이 필요하다.

이같은 문제점의 해결을 위해 본 논문에서는 기계적인 스위칭 시스템을 대신하고 터미날에서 직접 원하는 컴퓨터를 선택할 수 있도록 전자 교환기에서 사용하는 Time Division Multiplex switching 방식의 회로을 설계하여 이에 맞는 프로그램을 개발하고자 한다.

2. 본 론

다수의 컴퓨터와 terminal이 있는 경우 이들을 상호 접속 하기 위해서는 그림 1 과 같이 16 X 16 의 cross point 접속기 2 조가 필요하다. 그림 1 에서와 같이 terminal #1 과 computer #2가 서로 접속될 경우 X표로 된 2개의 접점을 연결시켜야 한다.

이를 위한 방법의 하나로써 한쪽의 연동 로터리 스위치에 연결하여 이것을 동시에 연결하는 방법이 있는데 이러한 방법은 많이 쓰이며 간단하기는 하나 사용자가 스위칭하는 곳으로부터 멀리 떨어져 있을 경우는 불편하고 그 수가 많을 때에는(예 16 X 16) 사용자의 실수로 하나의 컴퓨터에 두 개의 터미날을 연결하여 단락에 의한 RS-232 port 의 고장을 초래하는 일이 발생하고 오래 사용시 기계적인 접점의 산화로 접속이 잘 되지 않는 일이 발생한다. 또한 이것을 기계적 릴레이로 구성할 경우 16 X 16 - 256개의 2 접점 릴레이가 필요하며 이들을 구동및 관리하는 회로가 커지고 복잡하게 되며 기계적인 특성으로 인한 고장등에 대한 유지 보수가 곤란하다.

이러한 문제점의 보완을 위해 Time - Division 방식의 전자 교환기에서 사용하는 T -switch로 이러한 상호 연결을 할 수 있도록 그림 2 에 나타낸 블록과 같은 형태의 회로를 구성하였다.

2.1 회로의 구성

각 터미날 및 컴퓨터로 부터오는 Tx 신호는 multiplexer에서 하나가 선택되고 그것이 demultiplexer에서 원하는 목의 터미날 및 컴퓨터로 연결되게 된다. 이러한 순간적인 데이터의 연결은 입력 digital 신호의 baud rate의 2배 이상으로 반복되어 터미날 및 컴퓨터측에서는 항상 연결 된 것처럼 동작하게 된다. 실제에 있어 demultiplexer가 sequential write 그리고 multiplexer가 two-port RAM을 통해 그 데이터 출력에 따라 컴퓨터 및 터미날로 부터 오는 데이터를 random read 하는 sequential write-random read의 과정을 16개의 모든 channel 2 조에 대해 반복하게 된다.

timing generater로 부터의 address값이 two - port RAM을 통해 random read를 위한 데이터의 출력시 delay가 생기는데 그림 4에 timing chart를 나타냈다. 즉 demultiplexer에서 유효한 data를 write 하기 위해 multiplexer로 부터 오는 data가 충분히 안정된 값을 유지하고 있을 데이터 주기의

3/4의 시간에 sampling을 하도록 하고 있다. 따라서 clock rate는

2 X 9.6k baud-rate X 16 channel X 4 = 1.23 MHz의 약 2배인 2.4576 MHz로 하였다.

그리고 새로운 연결을 원할 경우에는 i8051의 컴퓨터로부터 입력을 받아 two - port RAM의 data를 바꾸어 주면 된다.

2.2 프로그램의 구성

i8051은 12MHz의 시스템 clock을 갖는 고속의 단일칩 컴퓨터로 RS -232 communication을 위한 hardware가 내장되어 있다. 이 serial port를 그림 1 에서의 COM #0에 연결하고 이 COM #0의 컴퓨터로 terminal의 각 Tx 선의 data 상태를 monitoring하여 새로운 연결을 위해 사용자가 누르는 특별한 ASCII characetr code (예, null character = CRTL-@) 를 감지하여 원하는 연결을 해주게 된다. 이것은 T - switching의 구조상 한 port의 source data가 여러 port로 read 될 수 있기 때문이다. 즉 연동식의 로터리 접점에서는 불가능한 single - source multiple - destination으로의 연결이 가능하다.

터미날 측에서 사용자가 special character인 CRTL-@을 반복해서 내 보낼 때 scanning에 의해 터미날 선의 상태를 monitoring하던 COM #0는 불완전할 지도 모르는 첫 character를 버리고 다음 character를 일정 시간 동안 기다려 CRTL-@ 일 경우 그 터미날에 message를 내 보내 사용자가 원하는 컴퓨터를 선택하도록 한다. 이러한 방식의 algorithm을 그림 5 에 abstract description으로 나타내고 i8051의 cross - assembler에 의해 coding 하여 i8051의 개발 장비인 SDK - 51에서 debugging 후 EPROM에 program하여 이것을 T - switching system board의 i8051용 EPROM socket에 끼워서 동작을 시켰다.

또 i8051은 9600 baud rate, 8-bit,no-parity 1-stop bit로 하였으므로 다른 터미날이 communication을 하고자 할 때 이와 같이 protocol을 일치 시켜야 한다.

3. 구성 결과

§2.1에서 설명한 바 대로 구성하여 그림 7 은 demultiplexer part의 파형을 보여 주고 있고, 그림 8 은 확대한 것으로 demultiplexer가 read time의 3/4 시간에 제대로 sampling 하도록 하는 신호가 나오고 있음을 보여 주고 있다. 그림 9는 COM #1 과 COM #4가 상호 연결된 경우 two - port RAM에서 나오는 신호를 잡은 것이다.

16 개의 scanning slot 중 각각 slot #1,slot #4에서 multiplexer가 select를 하도록 하기 위한 위한 신호를 정상적으로 발생하고 있음을 보여 준다. 또한 그림 10은 이 신호들을 시간축에 대해 확대하여 two-port RAM에서의 출력의 delay를 잡은 것으로 약 45 ns의 delay가 있음을 알 수 있다.

그림 6은 동작시 터미날에 나타나는 상태를 보인 것이다.

(1)은 사용자가 터미날 #7에서 CRTL-@ key를 누르고 있을 때 2 - 3 초 후 i8051 SCC 인 COM #0가 내 보내는 message이고,

(2)는 key 입력의 prompt가 나타나는 순간 SSM-16 computer의 TTY port 0를 선택하기 위해 숫자 5 를 친 후 터미날에 나타나는 상태이다.

(3)은 원하는 컴퓨터에 연결이 된 것을 확인 후 다음의 key 입력 prompt가 나타날 때 space-bar나 return key를 입력하여 i8051 SCC의 service를 빠져 나올 경우이다.

(4)는 실제로 SSM-16에 연결하여 login 후 동작하는 상태이다.

COM #0는 i8051에 연결되어 있어 COM #1 - COM #F 까지의 15 개가 사용 가능하며 사용자 편의를 위해 computer COM #F는 사용하지 않는 터미날의 parking용으로 하였다. 그럼에 함으로써 다른 사용자가 점유하지 않은 컴퓨터를 알수 있고 비어 있는 computer를 연결해서 사용할 수 있어 편리하기 때문이다.(그림 6 참조)

따라서 그림 6의 program에서는 터미날 16대 X 컴퓨터 14대의 상호 연결이 가능한 상태이다. 일반적으로는 15 X 15의 연결이 가능하다. 그 이유는 TTY #0및 COM #0는 자체 SCC에서 사용될 수 있기 때문이다.

4. 결 론

본 연구에서는 실험실이나 작업 현장에서 효과적으로 다중 컴퓨터및 다중 터미날을 접속하는 T-switching 시스템을 구현하였다.

이 장비는 완성되어 본 연구실에서 사용하고 있으며 앞으로 원격 제어를 위한 remote-data-logging system에의 응용이 가능하다.

5. 참 고 문 헌

1. Microcontroller Handbook, Intel corp. 1985
2. SDK - 51 USER'S MANUAL, Intel corp. 1983
3. V.E. BENES, Mathematical Theory of Connecting Networks and Telephone Traffic, Academic Press, 1965

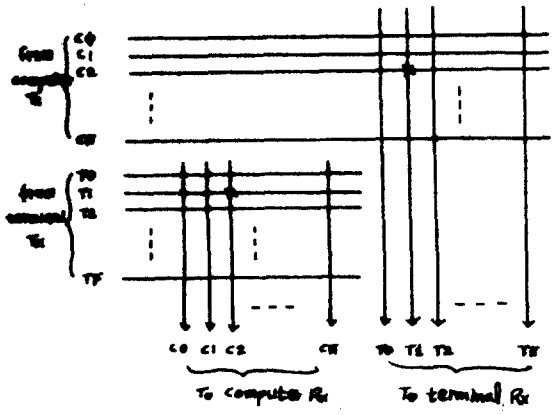

그림 1. 16개의 컴퓨터와 터미날이 상호 접속되기 위한 cross-point 접속도. x는 TERM#1 <--> COM#2가 접속될 경우를 나타낸다.

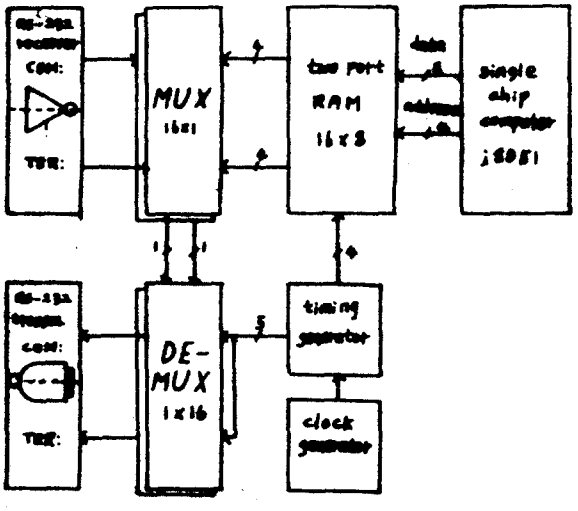

그림 2. 회로의 블록 선도

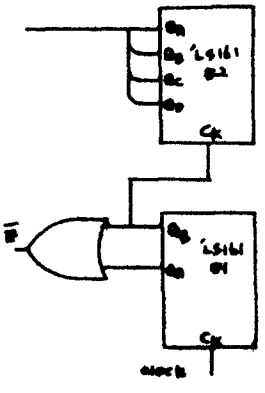

그림 3. timing generater의 회로

```
-------------------------------
abstract description of MAIN program
-------------------------------

for( TERM <- 1;    ; TERM <- (++TERM)/16 , term<>0 )
{
   disp <- no
   /* p1.0 .. p1.3 <-- TERM ! LED display of TERM */
   p1.4          <-- service or scanning_of_break
   (20+1) <-- 1h(=ser state)
   (20+F) <-- 2h /* brief status display for line printer */
   if ( TERM+20 )=ser then /* service status */
   {
        topen(TERM)      /* c0 --> Tn for message output */
        twinr(TERM.A.C)  /* dummy read  c0 <-- Tn */
        string ' an=your TTY#, type com#?[ '
        for ( r3-30.c :  r3<>0 & c : --r3 ) /* 1.5 sec waiting */
        {
              twinr(TERM.A.C)  /* key in waiting */
        }
        string backspaces.'wait]'
        if  not_time_out( i.e. c-0) then
        {
          if a-^] then
          {
             goto debugger
          }
          elseif  ( A= CR or space or escape ) then
          ( /* <--> quit to com# connection status
               thus no more prompt message          */
             TSW <-- TMAP
             ( TERM+20).0 <-- set scan status(1)
          )
          else      /* key_in */
          {
             if a-0..9 or A..F then
             (
                      Tn <-----> Cm
                /* Tnlow <- Cm. (C(Cpre)high <- T0) -> tsw
                   Cmhigh <- Tn.(T(Tpre)low  <- CF) -> tsw
                */
             )
             disp <- yes        /*     status display  ELSE10 */
          )
        )
   )
   else /* connection status */
   {
        twinr(.TERM.A.C )  /* break detection */
        if break( i.e. A-0 or ^P ) then
        {
          (TERM + 20).0 <- ser
          topen(TERM)
          disp <- yes
        )
   )
   if disp=yes then
   (   if (20+TERM).1 - 0 then
          if (20+TERM).1 - 0 then  status_display
          TTY ID display
   )
   TSW <-- TMAP
)
```

그림 5. 프로그램의 abstract description

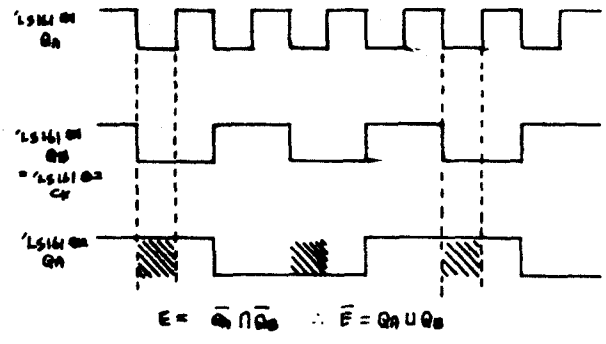

그림 4. 그림 3 의 회로의 timing선도

(1) 현재의 터미널 TTY #7 이 어느 컴퓨터에도 연결되지 않은 상태

```
***computer room products(86/mar/1-aug/15)***
*    WELCOME TO TDX SWITCHING SYSTEM@ajr     *
* 9600_BAUD,8_BIT,1_STOP_BIT,NO_PARITY       *
********(8051 acc communicates with you)******

## CONNECTION STATUS DISPLAY C(:T)<--T(:C) ##
   COM #1 ( VAX    ) <---    TTY #9
   COM #2 ( PDP8   ) <--- **idle**
   COM #3 ( PDP4   ) <--- **idle**
   COM #4 ( PDP5   ) <--- **idle**
   COM #5 ( SSM0   ) <--- **idle**
   COM #6 ( SSM1   ) <--- **idle**
   COM #7 ( SSM2   ) <--- **idle** |  (TTY#7): com# selection mode
   COM #8 ( SSM3   ) <---    TTY #3
   COM #9 ( DGU1   ) <---    TTY #6
   COM #A ( DGU2   ) <--- **idle**
   COM #B ( ZDS    ) <--- **idle**
   COM #C (        ) <--- **idle**
   COM #D (        ) <--- **idle**
   COM #E (        ) <--- **idle**
   COM #F ( park   ) <---    TTY #7

 for an connection type "F" ...TYPE COM# WHEN "(" APPEARS WITHIN "("
7 - your TTY#<-->COM# - F now#?(1-F)    (wait)

 /* here com # 5 is selected by typing "5" */
```

(2) prompt 순간 com# 5 를 선택한 경우의 상태

```
***computer room products(86/mar/1-aug/15)***
*    WELCOME TO TDX SWITCHING SYSTEM@ajr     *
* 9600_BAUD,8_BIT,1_STOP_BIT,NO_PARITY       *
********(8051 acc communicates with you)******

## CONNECTION STATUS DISPLAY C(:T)<--T(:C) ##
   COM #1 ( VAX    ) <---    TTY #9
   COM #2 ( PDP8   ) <--- **idle**
   COM #3 ( PDP4   ) <--- **idle**
   COM #4 ( PDP5   ) <--- **idle**
   COM #5 ( SSM0   ) <---    TTY #7
   COM #6 ( SSM1   ) <--- **idle**
   COM #7 ( SSM2   ) <--- **idle** |  (TTY#7): com# selection mode
   COM #8 ( SSM3   ) <---    TTY #3
   COM #9 ( DGU1   ) <---    TTY #6
   COM #A ( DGU2   ) <--- **idle**
   COM #B ( ZDS    ) <--- **idle**
   COM #C (        ) <--- **idle**
   COM #D (        ) <--- **idle**
   COM #E (        ) <--- **idle**
   COM #F ( park   ) <--- **idle**

 for an connection type "F" ...TYPE COM# WHEN "(" APPEARS WITHIN "("

7 - your TTY#<-->COM# - 5 now#?(1-F)    (wait)

/* here space bar is pressed for exit from tdx selection mode */
```

(3) space bar에 의해 빠져나왔을 때의 상태 message

```
5 - COM# <---------->TTY# - 7 /* connection message */
```

(4) message후 실제로 SSM-16에 연결된 상태

```
login: ajr/* prompt from SSM-16 */
Password:

      *****  Last login : Tue Sep  9 22:48:18 1986 on tty5  *****

            SST SSM-16 "NORA" system with CONIX (UNIX V.7)
            Welcome to KAIST Control Lab. 86.07.08

./          ../         .login*     core*       core~       dum.fnt     edge.s88
edge.c      edge.o      ker/        kkk.c       load.c*     mount*      sref.del
signal.h    str/        t2.dat      test/       test.f77    test.for    test.o
test.obj    test.p
[ajr:]]]a
```

그림 6. 동작시 터미널에 나타나고 있는 상태

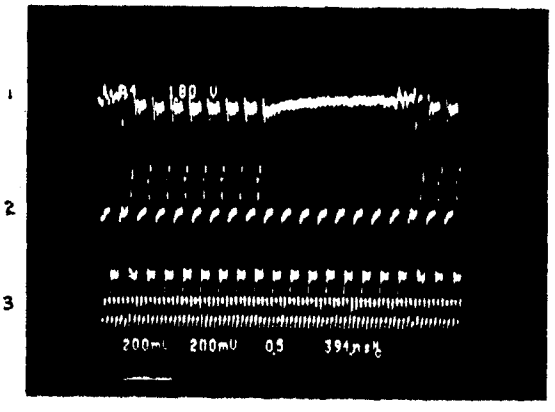

그림 7. 'demultiplexer part의 파형. 2개의 'LS259가 8개씩 demultiplex하고있다. (1) 'LS259의 /E input. (2) 'LS161#2의 Ck input. (3) 'LS161#1의 Qa output

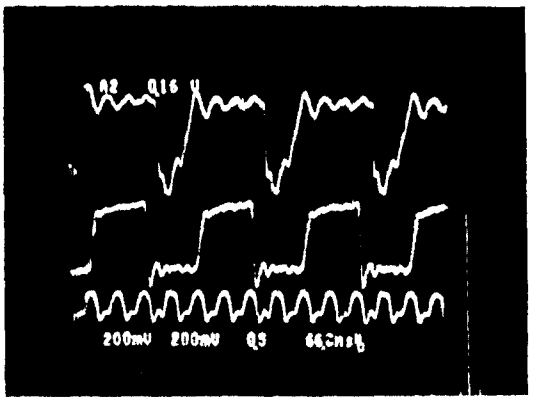

그림 8. 그림 7을 확대한 상태. 'LS161#2의 Ck의 한 주기의 3/4의시간에 'LS259의 /E가 제대로 enable되고 있음을 보여주고있다. (1),(2),(3)은 그림 7과같다.

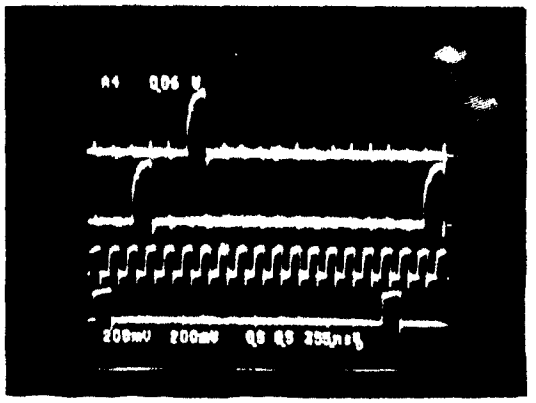

그림 9. COM#1과 TTY#4가 상호 연결된 상태에서 time slot 1과 4에서 high pulse가 나타나고있다.
(1) two-port RAM의 Q3, (2) two-port RAM의 Q5,(3) 'LS161#2의 ripple out

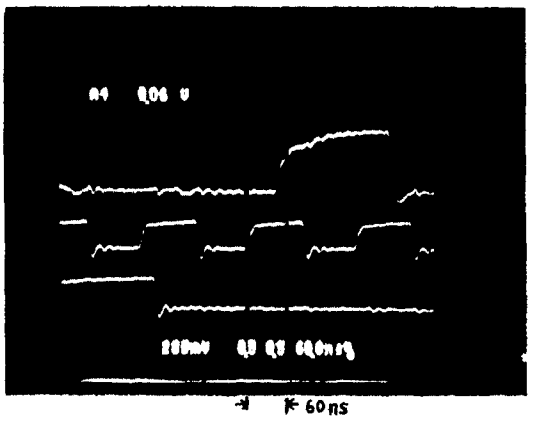

그림 10. 그림 9를 확대한것. 'LS161#2의 Ck로부터 two-portRAM의 Q5가 high로 될때까지 약 45ns의 delay가 있음을 알수있다. (1),(2),(3)ms 그림 9와같다.