

## An RS232-RS485 Communication Multiplexer

### **Abstract**

This project presents a solution to provide 8 RS-485 communication ports using one RS232-RS485 conversion circuit. 80HC908QY1 controls 8 analog switches and builds connections between real RS-485 communication port and any one of the 8 virtual ports. User can turn on the target channel manually, or let the MCU control the channels automatically, or control the channels by a special designed application program running on the PC. All the working statuses can be displayed on a LCD. All the working parameters can be saved in the Flash without worrying about power failure.

This device is handy to operate with only two buttons and an easy reading LCD. Pressing the Fkey, user can select one of the 6 function modes, which are hand controlling, auto scanning, PC co-working, maximum scan channel setting, scan interval setting, and parameters saving. In one function mode, Skey completes different tasks, such as channel changing, parameters setting, data saving and etc. A LCD lets user know the working status, e.g. the current selected channel number. It also helps user to operate the device with some prompts.

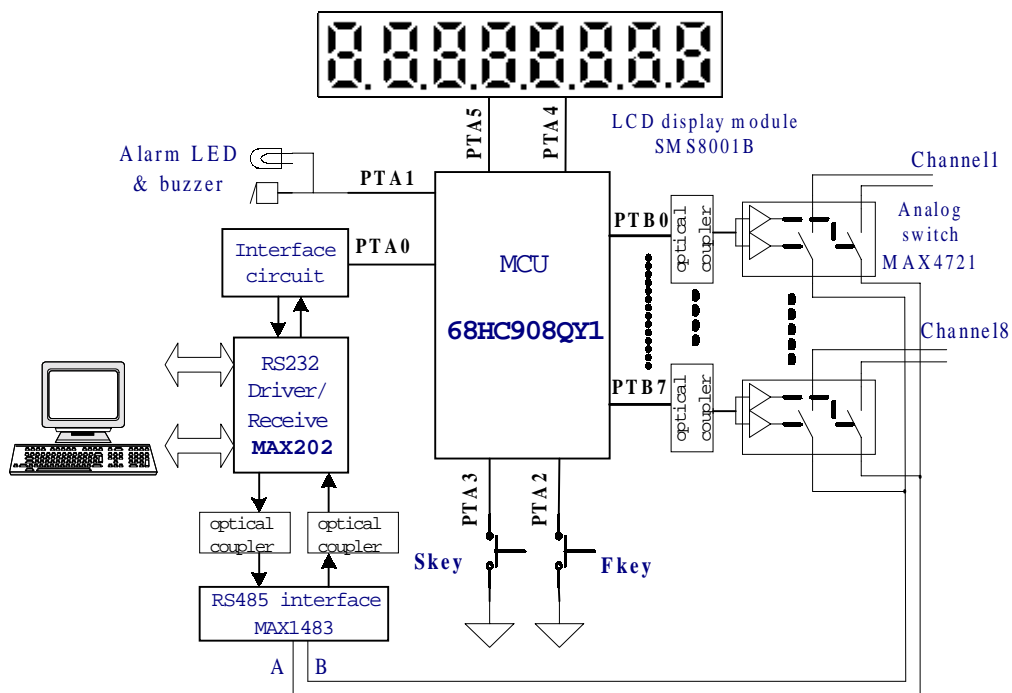
Most of the features of 80HC908QY1 are dug out in this application, such as key board interrupt for button handling, powerful I/O pins for analog switches and LCD controlling, flash for parameters saving, low power mode software system for energy saving, and etc. The maximum channel number and scan interval can be set by user and saved in the 908QY1's flash. Through PTA0, this device can communicate with PC. In the PC connection mode, it can receive the commands from PC and execute these commands to control the 8 channels. Additionally, some subroutines in 908QT4's ROM are used to simplify the program.

The software is driven by keyboard and timer overflow interrupts. When the 908QT is wakened up by interrupts, it does the routine work of the selected function mode. After that, the MCU goes to sleep again. Therefore, in most of the time the MCU stays in low power mode (WAIT).

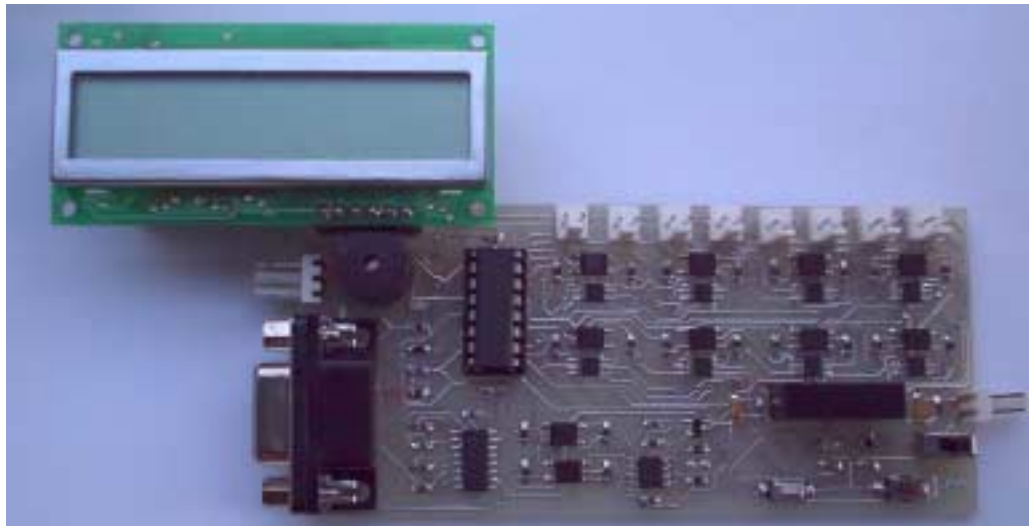
The control circuit and the RS232 interface circuit are insulated from the RS-485 interface circuit and the 8 analog switches. This is an effective method to obtain high reliable communication. A DC-DC converter and some optical couplers are employed to insulate these two parts. With DC-DC converter, two separate power supply can be gained through one power supply input. Accordingly, only single 5V supply can power this multiplexer. Moreover, it is tested that a 3V battery can make it run.

This multiplexer is useful to the industrial application and many other applications. Its delicate structure and PC based communication solution make it easy to be integrated into a computer controlled application system.

**Block Diagram**



*Photos of this project*





**Software code samples**

```

;*****
;*      Flsh_Erase_Page: Erase a flash page
;*      Entry parameter table
;*      1)      H:X          The any address in the erasing flash page
;*      2)      $0088       The control byte address in RAM
;*                                     if the 6th bit =1 : erase all
;*                                     if the 6th bit =0 : erase a flash page
;* A subroutine (ERAGNGE) in ROM is invoked
;* The entry address is $2A71 (verified in 908QY1, 908QT4)
;*****
Flash_Erase_Page:
    LDHX    #FPARASADD+2        ; The any address in the erasing page
    BCLR   Bit6,FlashEraseCon   ; Erase a pages, FlashEraseCon --$88 in ROM
    JSR    ERARNGE             ; Call the subroutine in the ROM
    RTS

;*****
;*Open_A_Channel: open a target channel. The channel number is stored in
;*                X, based on 1. This sub routine save the turn on signal (zero) in
;*                Carry bit. Rotate left through Carry times, as X determined.
;* Input :       X
;* Output:       PORTB
;*****
Open_A_Channel:
    LDA    #$FF                ; Load all turn off signals
    CLC                                ; Clear C as the turn on signal
OAC_Next:
    ROLA   ; Rotate left through Carry
    DBNZX OAC_Next ; The target bit is cleared?
    STA    PORTB ; Send result to the I/O port
    RTS

;*****
;* Hex_2_ASCII: Convert a byte from Hex format into ASCII format
;* Input : A
;* Output: TempASCII -- the low byte of the result
;*                TempASCII+1 -- the high byte of the result
;*****
Hex_2_ASCII:
    PSHA   ; Save A in the stack
    AND    #$0F ; Get 4-low bits
    ADD    #$30 ; Convert to ASCII based on number '0'
    CMP    #$3A ; Larger than "10"?
    BLT    H2A_next ; Less than "10" covert success
    ADD    #$07 ; Add extra 7 for ASCII conversion
H2A_next:
    STA    TempASCII ; Save the converted result
    PULA   ; Reload the Hex
    LSRA
    LSRA
    LSRA
    LSRA ; Get 4-high bits
    ADD    #$30 ; Convert to ASCII based on number '0'
    CMP    #$3A ; Larger than "10"?
    BLT    H2A_over ; Less than "10" covert success
    ADD    #$07 ; Add extra 7 for ASCII conversion
H2A_over:
    STA    TempASCII+1 ; save the converted result

```

RTS

```

;*****
;* Hex_2_BCD: Convert a byte from Hex form into BCD form.
;* Input :      X -- the converting byte X should be smaller than 100
;* Output:      A
;*****
Hex_2_BCD:
    CLRH
    CLRA                                ; Initialize the sum register
H2B_Next:
    ADD      #$01
    DAA
    DBNZX H2B_Next                      ; Add next step
    RTS

;*****
;* Function_Routine: Complete the routine work in each function mode
;* When the MCU is wakened up by the interrupts, this routine would be run at once
;* All the subroutine in this routine is grouped by the function.
;* The "Index" indicates which functional mode the program should be entered
;*****
Function_Routine:
    CLRH
    LDX      Index                      ; Load the Index number
    LSLX                                ; X = X * 2
    JMP      Function_Branch, X
Function_Branch:
    BRA      Fun1
    BRA      Fun2
    BRA      Fun3
    BRA      Fun4
    BRA      Fun5
    BRA      Fun6
Fun1:      ; Function mode ( Manual Control )
    JSR      Manual_Control ; Control the channel manually
    BRA      FR_over
Fun2:      ; Function mode ( Auto Scan )
    JSR      Auto_Scan      ; Scan & control channels form 1- max
    LDA      ScanInterval   ; Load the scan interval timeout counter
    STA      Timeout1H     ; Load n, all timeout value = n*102.4ms
    MOV      #20,Timeout1  ; Load the 102.4ms timeout value
    BRA      FR_over
Fun3:      ; Function mode (communicate with PC via PTA0 )
    BCLR    FKEY,Kflag     ; Clear the Fkey pressed mark
    JSR      Serial_Communication
    BRA      FR_over
Fun4:      ; Function mode ( Max Channel set)
    JSR      Max_Channel_Set
    BRA      FR_over
Fun5:      ; Function mode ( Scan Interval set)
    JSR      Interval_Set
    BRA      FR_over
Fun6:      ; Function mode ( Parameters Save)
    JSR      Parameters_Save
    BRA      FR_over
FR_over:
    RTS

```