

HT48 & HT46 MCU UART Software Implementation Method

D/N : HA0004E

Introduction

With low power consumption and high performance, Holtek's 8-bit microcontrollers are suitable for a variety of system control product applications including office automation and various consumer products.

Sometimes in some application systems, there's a need to have a serial asynchronous interface with other microcontrollers. In such cases, UART Software implementation is useful for applications requiring cost-effective method of low throughput data transmission between multiple devices, compared to any special-purpose IC hardware method. This Application Note introduces a simple implementation method of a software-simulated universal asynchronous receiver transmitter (UART). The software UART supports the basic 8-N-1 format, which is 8 data bits, no parity, and 1 stop bit. In serial data communication, data is transmitted sequentially; one bit at a time, and the transfer rate can be adjusted depending on the system frequency.

Operating Description

This section describes the procedure to test the UART Software using the `descript.inc` and `dfseript.asm` files. When testing, refer to the following steps:

- Basing on the actual circuitry, define the TXPIN and the RXPIN in the `dfseript.inc` file, separately as data transmission and reception pins.
- For serial data transmission, the system is initially configured to a certain baud rate by setting the baud rate constant.

$$\text{Baud Rate Constant} = [\text{System frequency } f_{\text{CLK}} / (\text{baudrate} \times 12)] - 3$$

Or, the baud rate constant can also be based on the following table:

System Frequency Data Transfer Speed (bps)	4MHz	2MHz	1MHz
9600	31	14	X
7200	43	20	8
4800	66	31	14
3600	89	43	20
2400	135	66	31
2000	163	80	38
1800	182	89	43
1200	X	135	66

X : not applicable

After determining the baud rate constant, modify the value defined in the `dfseript.inc` file.

- The contained files `dfseript.inc` and `dfseript.asm` enhanced the transmission and reception execution functions in the original file.
- Execution of the two functions requires the following conditions:
 - Transmit: before calling the 8-bit data to be transmitted, they must be placed in the emulator
 - Receive: the 8-bit data received must be placed in the emulator

Example

Suppose we're given the following conditions:

1. System frequency: 4MHz
2. Serial transmission speed: 4800
3. Use PA3 as transmitting pin
4. Use PA2 as receiving pin

```

;-----
;File name: DFSERIPT.INC
;Author: Jason Tseng
;-----
BaudRateConst EQU 66          ;can be taken from the table
TXPIN EQU PA.3
RXPIN EQU PA.2
ifndef DFSERIPT_ASM

```

```
EXTERN TRANSMIT : NEAR
EXTERN RECEIVE : NEAR
endif
;-----
; End of DFSERIPT.INC
;-----
;
;File name: TEST.ASM
;Author: Jason Tseng
;-----
include dfscript.asm
MyData .SECTION 'DATA'
:
MyCode .SECTION 'CODE'
ORG 0H
JMP START
START :
:
SET PAC.2 ;set the receiving pin as input
CLR PAC.3 ;set the transmitting pin as output
:
MOV A, TRANSMIT_DATA ;data to be transmitted must pass
;through the accumulator ACC
CALL TRANSMIT ;transmit the subroutine call
:
:
CALL RECEIVE ;receive the subroutine call
MOV RECEIVE_DATA, A ;receive data from the accumulator ACC
:
END
;-----
; End of TEST.ASM
;-----
```

System Resource Options

RAM space: 4
ROM space: 49
I/O pin: 2

Appendix

This Application Note provides two files: dfseript.inc (for details, refer to Appendix A) and dfseript.asm (for details, refer to Appendix B). The Software UART implemented in this Application Note supports the most common UART protocol, 8-N-1, so we assume that the data transmission flow consist of 8 bits data, no parity check and 1 stop bit. Basically, we have to expect that this will significantly consume MCU processing power and that there will be some errors or design constraints in using a software program to simulate hardware functionality. Generally speaking, the higher the system frequency and the baud rate, the smaller is the error. Basing from our experience, the best value for the baud rate constant is between 7~256.

Appendix A

```
-----  
; File name: DFSERIPT.INC  
; Author: Jason Tseng  
; Objective: Set the applicable baud rate constant, TXPIN and RXPIN  
include ht48r30a-1.inc  
BaudRateConst EQU 66  
TXPIN EQU PA.3  
RXPIN EQU PA.2  
ifndef DFSERIPT_ASM  
EXTERN TRANSMIT : NEAR  
EXTERN RECEIVE : NEAR  
endif  
-----  
; End of DFSERIPT.INC  
-----
```

Appendix B

```
-----  
; File name: DFSERIPT.ASM  
; Author: Jason Tseng  
-----  
#define DFSERIPT_ASM  
#INCLUDE DFSERIPT.INC  
PUBLIC TRANSMIT  
PUBLIC RECEIVE  
BAUDRATE EQU BAUDRATECONST ;baud rate constant  
; (in dfseript.inc, user defined)  
TX EQU TXPIN ;user defined TXPIN pin, TX  
RX EQU RXPIN ;user defined RXPIN pin, RX  
SDATA .SECTION 'DATA'  
COUNT DB ? ;serial bit counter  
TXREG DB ? ;transmit to data register  
RCREG DB ? ;receive from data register
```



```
DELAY DB ? ;delay counter
SERIAL .SECTION 'CODE'
TRANSMIT PROC ;TRANSMIT subroutine
MOV TXREG,A ;transmit data to TXREG
MOV A,BAUDRATE ;set the baud rate delay
MOV DELAY,A ;
CLR TX ;transmit start flag bit "0"
MOV A,9 ;set transmit data bit number
MOV COUNT,A ;
TXDELAY: ;
SDZ DELAY ;baud rate delay loop
JMP TXDELAY ;
MOV A,BAUDRATE ;reload baud rate delay constant
MOV DELAY,A ;
SDZ COUNT ;check if data transmission is finished
JMP SENDBIT ;if transmission is not finish, jump to
;SENDBIT to transmit next bit

JMP ENDTX ;if data transmission is finished,
;jump to ENDTX

SENDBIT:
RRC TXREG ;rotate right through C to get txreg
;transmitted data
SNZ C ;check if there's a borrow at C
JMP LOBIT ;C=0 jump to LOBIT transmit "0"
SET TX ;C=1 transmit "1"
JMP TXDELAY ;
LOBIT: ;
CLR TX ;transmit "0"
JMP TXDELAY ;
ENDTX: ;
NOP ;no operation
NOP ;
SET TX ;stop transmitting bit
T1: ;
SDZ DELAY ;delay between each bit transmission
;period

JMP T1 ;
MOV A,BAUDRATE ;
MOV DELAY,A ;
T2: ;
SDZ DELAY ;
JMP T2 ;
RET ;
TRANSMIT ENDP ;
RECEIVE PROC ;receive subroutine
SZ RX ;check start bit "0"
JMP RECEIVE ;if there's no start bit, jump to
;receive
MOV A,9 ;receive start bit initialization data
```



```
MOV COUNT,A                ;set received bit number
MOV A,BAUDRATE+1           ;set baud rate delay
MOV DELAY,A                ;
RXDELAY:                   ;
SDZ DELAY                  ;baud rate delay
JMP RXDELAY                ;
MOV A,BAUDRATE+1           ;reload baud rate constant
MOV DELAY,A                ;
SDZ COUNT                  ;check if finished receiving data
JMP RXBIT                  ;if receiving is not finish, jump to
                            ;RXBIT receive next bit data
MOV A,RCREG                ;if finish receiving, prepare to
                            ;receive data
RET                         ;
RXBIT:                     ;
SET C                      ;set received bit to "1"
SNZ RX                      ;check if received bit is "1"
CLR C                      ;if receive flag is "0", clear received
                            ;bit to "0"
RRC RCREG                  ;rotate through right of rcreg
JMP RXDELAY                ;
RECEIVE ENDP              ;
END
;-----
; End of DFSERIPT.ASM
```