Joy of innovation
nuvoTon

SPI-to-UART

User Manuel

www.nuvoton.com

Overview

- **Introduction**
  - Feature List
  - Multi Function Pin Setting
  - SW Protocol Layer
- **API Command**
  - API Flow Chart
  - API Command List
- **Sample Code**
- **SPI Command**
  - SPI-to-UART Translator Protocol
  - Packet Format
  - Instruction Set Table

Introduction

- **N9H20 expand two group of UART via Mini58 built-in UART**
- **UART can be configured and controlled via SPI**

Related configure
SPI0_SCLK:8M

Feature List

- **SPI x 1**
  - Default clock rate set by firmware
- **UART x 2**
  - Default 115200 bps, 8-bit, non-parity, 1-stop
- **Performance**
  - UART0 baud rate can be up to 115200 bps
  - UART1 baud rate can be up to 57600 bps

Multi Function Pin Setting

N9H20

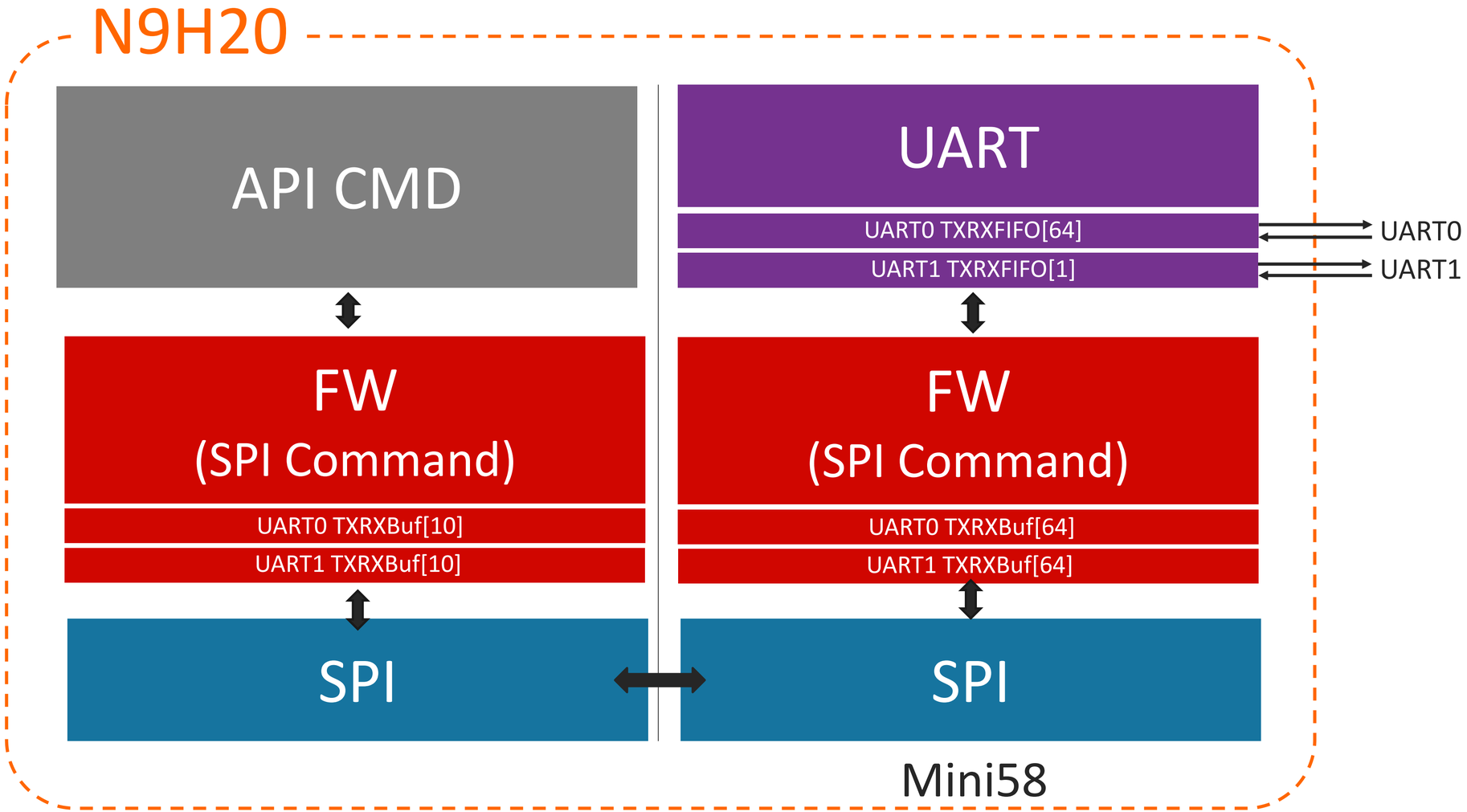
Pin No.	Function	Pin No.	Function
MF_GPD15	SPI0_DO	MF_GPD14	SPI0_DI
MF_GPD12	SPI0_CLK	MF_GPA5	SPI0_CS1_
MF_GPA4	GPIOA[4] ◆		

◆ Interrupt

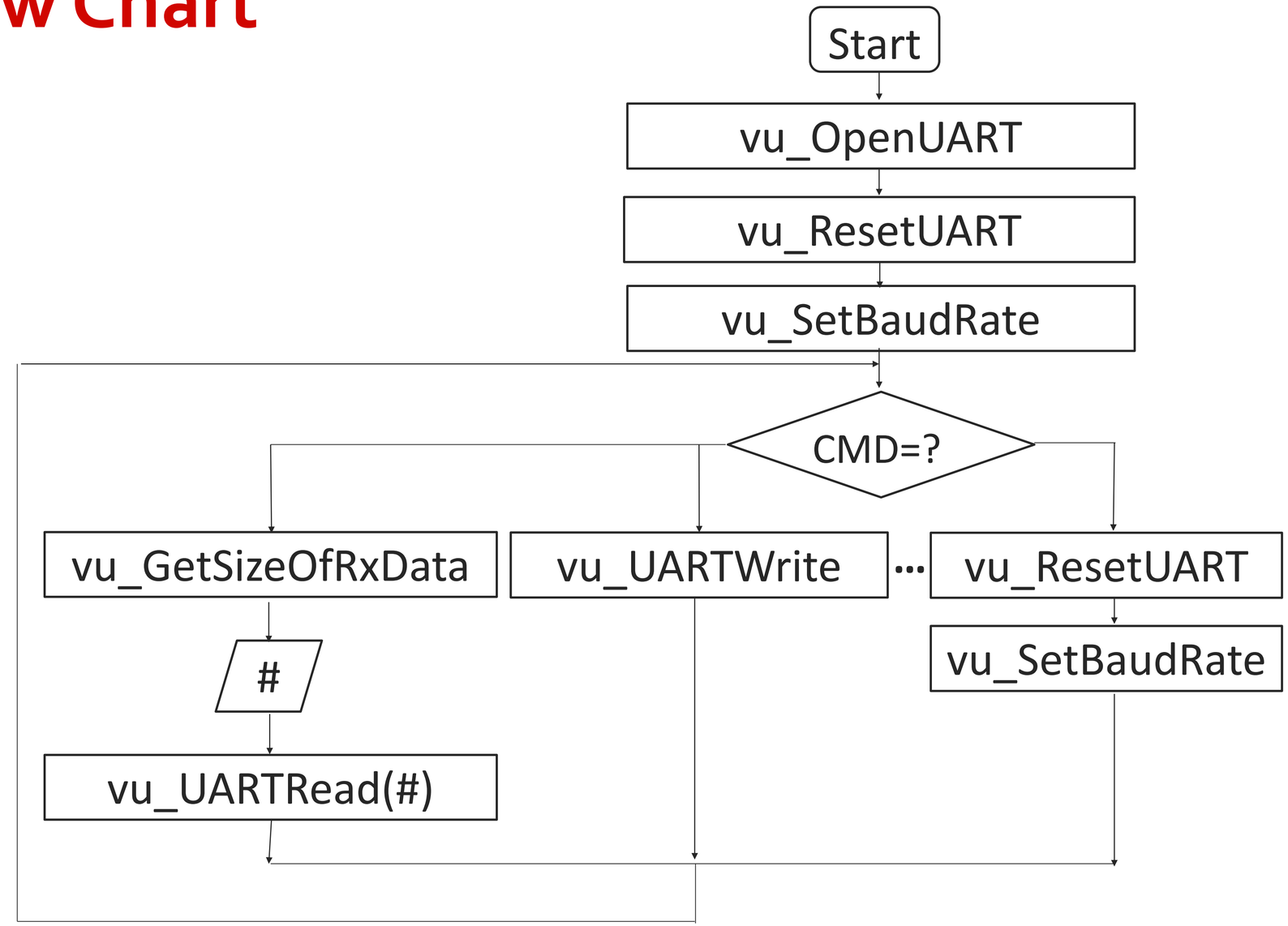
Mini58

Pin No.	Function	Pin No.	Function
SYS_MFP_P12	UART0_RXD	SYS_MFP_P13	UART0_TXD
SYS_MFP_P24	UART1_RXD	SYS_MFP_P25	UART1_TXD
SYS_MFP_P04	SPI0_SS	SYS_MFP_P05	SPI0_MOSI
SYS_MFP_P06	SPI0_MISO	SYS_MFP_P07	SPI0_CLK
SYS_MFP_P14	GPIO ◆		

SW Protocol Layer



API Flow Chart



schematic diagram

@vu: virtual UART

API Command List

- vu_OpenUART
- vu_ResetUART
- vu_SetBaudRate
- vu_UARTWrite
- vu_UARTRead
- vu_GetRXAvailLen
- vu_GetTXFreeLen
- vu_GetNotification
- vu_ClearBuf
- vu_CloseUART
- vu_GetStatus

vu_OpenUART

- **vu_OpenUART(UINT8 UART_port)**
  - UART_port
    - UART_PORT0 0x0
    - UART_PORT1 0x1
    - UART_ALL 0xff
- **Example : vu_OpenUART(UART_PORT0)**
- **Include :**
  - Enable GPIO_INT, SPI, UART

vu_ResetUART

- **void vu_ResetUART(UINT8 UART_port)**
  - Parameter (UART_port)
    - UART_PORT0 0x0
    - UART_PORT1 0x1
    - UART_ALL 0xff
    - RST_SPI 0xfe //for N9H20 SPI RST
- **Example**
  - vu_ResetUART(UART_PORT0)
- **Include**
  - Clear RXTXBuf, RXTXFIFO
  - Reset UART IP

vu_SetBaudRate

- **void vu_SetBaudRate(UINT8 UART_port, UINT32 baudrate)**
  - Parameter (UART_port)
    - UART_PORT0 0x0
    - UART_PORT1 0x1
  - Example
    - vu_ResetUART(UART_PORT0,115200)
  - Set two group of UART:
    - Baud rate need to be separated to set
    - UART1 baud rate can be up to 57600 bps when UART0 115200 bps

vu_UARTWrite

- **void vu_UARTWrite(UINT8 UART_port, unsigned char *pSrc, INT32 len)**
  - Parameter (UART_port)
    - UART_PORT0 0x0
    - UART_PORT1 0x1
- **Example**
  - vu_UARTWrite(UART_PORT0, pSrc,1)

vu_UARTRead

- **int vu_UARTRead (UINT8 UART_port,INT32 Max,UINT8 *pDst)**
  - Parameter (UART_port)
    - UART_PORT0 0x0
    - UART_PORT1 0x1
  - Example
    - # = vu_UARTRead (UART_PORT0,1, pDst) // return amount of data user read

vu_GetRXAvailLen

- **int vu_GetRXAvailLen (UINT8 UART_port)**
  - Parameter (UART_port)
    - UART_PORT0 0x0
    - UART_PORT1 0x1
    - UART_ALL 0xff
  - Example
    - # = vu_GetRXAvailLen(UART_PORT0)//return amount of data in RX buffer
  - UART_ALL
    - # = vu_GetRXAvailLen(UART_ALL)
    - [0:3]# : UART0
    - [4:7]# : UART1

vu_GetTXFreeLen

- **int vu_GetTXFreeLen(UINT8 UART_port)**
  - Parameter (UART_port)
    - UART_PORT0 0x0
    - UART_PORT1 0x1
    - UART_ALL 0xff
  - Example
    - # = vu_GetTXFreeLen (UART_PORT0)//return length of TX buffer could be put
  - UART_ALL
    - # = vu_GetTXFreeLen(UART_ALL)
    - [0:3]# : UART0
    - [4:7]# : UART1

vu_GetNotification

- **char vu_GetNotification(void)**
  - Example
    - status = vu_GetNotification()
  - status
    - 0x1 = has data in UART0 RX buf
    - 0x2 = UART0 TX buf is empty
    - 0x4 = has data in UART1 RX buf
    - 0x8 = UART1 TX buf is empty
  - be defined
    - #define VUART_TX_EMPTY(port)
    - #define VUART_RX_INT(port) //rx interrupt

vu_ClearBuf

- **void vu_ClearBuf(UINT8 UART_port)**
  - Parameter (UART_port)
    - UART_PORT0 0x0
    - UART_PORT1 0x1
    - UART_ALL 0xff
  - Example
    - vu_ClearBuf(UART_PORT0)
  - Include
    - Clear RXTXBuf, RXTXFIFO

vu_CloseUART

- **void vu_CloseUART(UINT8 UART_port)**
  - Parameter (UART_port)
    - UART_PORT0 0x0
    - UART_PORT1 0x1
    - UART_ALL 0xff
    - CLOSE_SPI 0xfe
  - Example
    - vu_CloseUARTPort(UART_PORT0)

vu_GetStatus

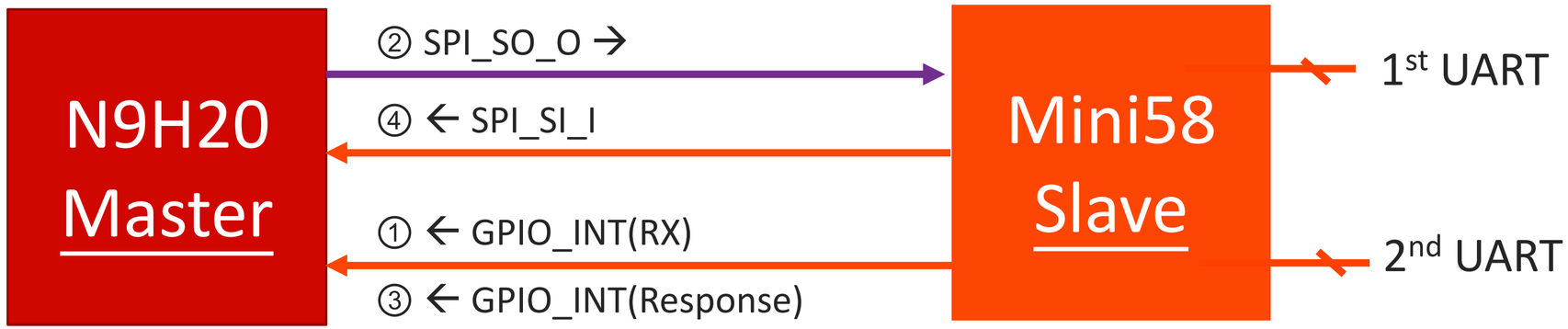
- **int vu_GetStatus(void)**
  - Example
    - status = vu_GetStatus()
  - status
    - 1 = UART is busy to do last CMD
    - 0 = UART is free and can execute new CMD

Sample Code

```
 //wait  mini58 response package
while(vu_GetStatus()==1);
//wait rx interrupt //SPI_UART=UART_PORT0
while(VUART_RX_INT(SPI_UART))
{
    //ask how much data is in rx buffer
    tmp4=vu_GetRXAvailLen(SPI_UART);
    //read  data and return practical amount of  data user read
    tmp2=vu_UARTRead(SPI_UART, tmp4,pDst)
    //write  # bytes data to TX buffer // pDst: source address
    vu_UARTWrite(SPI_UART,pDst,tmp2);
}
```

SPI-to-UART Protocol

- ① Mini58 will generate GPIO Interrupt to N9H20 after Mini58 receives data from UART RX
- ② If N9H20 UART buffer (maximum size : 10) is not full, N9H20 will do ‘Receive’ command through SPI_SO_O
- ③ When Mini58 has finished task, it will generate GPIO Interrupt to informs N9H20 that it has finished command and asks N9H20 to receive response package from SPI_SI_I
- ④ N9H20 receives response package from SPI_SI_I
- ② → ③ → ④ for other commands



Packet Format

- **Command packet**

Offset	0	1	2	3	4..n
Content	‘C’	Checksum	Length	Command	Parameters (or Data)

- **Status packet**

Offset	0	1	2	3	4..n
Content	‘R’	Checksum	Length	Command	Status(or Data)

- **Checksum : Checksum of length, status, and data. (Unsigned 8-bit calculation)**
- **Length : Length of status and data.**
- **Status : Status depend on previous command.(0:ok)**

SPI Command List

- OpenUARTPort
- ResetUARTPort
- ConfigureUARTPort
- UARTNotification
- QueryUARTTXBuffer
- SendDatatoUARTTXBuffer
- QueryUARTRXBuffer
- ReceiveDatafromUARTRXBuffer_Max
- CloseUARTPort
- ClearBuf

Instruction Set Table

CMD	Package	Byte							
		0	1	2	3	4	5	6	7
OpenUARTPort	CMD	0x43	CKSM	0x2	0xA	Port	N/A		
	Status	0x52	CKSM	0x2	0xA	0	N/A		
ResetUARTPort	CMD	0x43	CKSM	0x2	0x0	Port/▲	N/A		
	Status	0x52	CKSM	0x2	0x0	0	N/A		
ConfigureUARTPort ※	CMD	0	1	2	3	4	5	6	7
		0x43	CKSM	0x9	0x1	Port0/1	baud	baud	baud
		8	9	10	11	12	13	14	15
		baud	Data Width	Parity	Stop	N/A			
	Status	0x52	CKSM	0x2	0x1	0	N/A		
UARTNotification	CMD	0x43	CKSM	0x2	0x3	Port	N/A		
	Status	0x52	CKSM	0x2	0x3	0	N/A		

Port: UART_PORT0/UART_PORT1/UART_ALL

▲: RST_SPI

※: Data width, parity, stop haven't be implemented

Instruction Set Table

CMD	Package	Byte							
		0	1	2	3	4	5	6	7
QueryUARTTXBuffer	CMD	0x43	CKSM	0x2	0x4	Port	N/A		
	Status	0x52	CKSM	0x3	0x4	Port0	Port1	N/A	
SendDatatoUARTTXBuffer	CMD	0	1	2	3	4	5…		
		0x43	CKSM	n-2	0x5	Port0/1	Data		
		…n							15
		Data							N/A
	Status	0x52	CKSM	0x2	0x5	0	N/A		
	CMD	0	1	2	3	4	5…		
		0x43	CKSM	◎	0x5	ALL	PORT0 Data		
		…n		10…m					15
		PORT0 Data		PORT1 Data					N/A
	Status	0x52	CKSM	0x2	0x5	0	N/A		

n : transmit n bytes data to UART PORT0

m: transmit m bytes data to UART PORT1

◎ **:** ((m<<4)|n)

Instruction Set Table

CMD	Package	Byte							
		0	1	2	3	4	5	6	7
QueryUARTRX Buffer	CMD	0x43	CKSM	0x2	0x6	Port	N/A		
	Status	0x52	CKSM	0x3	0x6	Port0	Port1	N/A	
ReceiveDatafromUARTRX Buffer_Max	CMD	0x43	CKSM	0x3	0x7	Port0/1	n	N/A	
	Status	0	1	2	3	4…			
		0x43	CKSM	n-1	0x7	Data			
		…n						14	15
		Data						N/A	
	CMD	0x52	CKSM	0x3	0x7	ALL	n	m	N/A
	Status	0	1	2	3	4…			
		0x43	CKSM	◎	0x7	Port0 Data			
		…n	9…m					14	15
		Port0 Data	Port1 Data					N/A	

n : receive n bytes data from UART Port0

m: receive m bytes data from UART Port1

◎ **:** ((m<<4)|n)

Instruction Set Table

CMD	Package	Byte							
		0	1	2	3	4	5	6	7
CloseUARTPort	CMD	0x43	CKSM	0x2	0x9	Port/▲	N/A		
	Status	0x52	CKSM	0x3	0x9	0	N/A		
ClearBuf	CMD	0x43	CKSM	0x2	0xA	Port	N/A		
	Status	0x52	CKSM	0x2	0xA	0	N/A		

▲: CLOSE_SPI