

# SA818 Programming Manual

Standard Uart interface is used to configure the parameter of SA818 Walkie Talkie. The format of UART is 9600, 8, N, 1, which means: Baud = 9600, data bit = 8bit, Parity = None, Stop bit = 1 bit. The interface is TTL or CMOS @3.3V. The parameters which can be configured are: Bandwidth, Tx-frequency, Rx-frequency, SQ, Tx\_SubAudio(Tx\_CTCSS/Tx-CDCSS), Rx\_SubAudio (Rx\_CTCSS/Rx-CDCSS), Volume, Scan frequency, Audio filter, etc. When commands received, the module will verify and reply acknowledge message; User should wait enough time to send the next command until received the acknowledge signal.

## 1. Instruction Command Format:

All instructions command ended with <CR><LF>, and ASCII is used.

## 2. Commands List:

There are 5 commands in total to be used, below is the list:

- Command 1: Handshake instruction command
- Command 2: Group parameters configuration
- Command 3: Scan frequency configuration
- Command 4: Volume configuration
- Command 5: Audio filter command

### **3. Instruction Command Description:**

#### **3.1) Handshake instruction command**

Format: AT+DMOCONNECT <CR><LF>

Reply: +DMOCONNECT:0<CR><LF>

#### **3.2) Group parameters configuration**

Format: AT+DMOSETGROUP=BW, TX\_F, RX\_F, Tx\_subaudio, SQ, Rx\_subaudio

Sample 1: AT+DMOSETGROUP=0,415.1250,415.1250,0012,4, 0013

Sample 2: AT+DMOSETGROUP=0,415.1250,415.1250,754N,4, 445I

Parameter Description:

BW: 0: 12.5K 1: 25K

TX\_F: Tx Frequency value, SU818-U: 400~480MHz,  
SU818-V :134~174MHz,

RX\_F: Rx Frequency value, SU818-U: 400~480MHz,  
SU818-V :134~174MHz,

Tx\_subaudio: Tx\_CTCSS or TX\_CDCSS

Rx\_subaudio: Rx\_CTCSS or RX\_CDCSS.

If 0000 is set, that means subaudio function is disable. No CTCSS and no CDCSS.

The range of Tx\_CTCSS and Rx\_ctcss is 1—38.

Please find below table for Tx\_CDCSS and Rx\_CDCSS. The table is only list the code ended with “I”, the code ended with “N” is not shown. Actually the code ended with “N” can be used without any problems.

The code ended with “N” is the complement of the code ended with “I”.

For example:

023I = 11001000000111000110111

023N=~023I=00110111111000111001000

SQ: 0 ~ 8

0: listen mode

1—8: Different SQ Level.

### 3.3) Scan frequency configuration

Purpose: Check if the matched signal exist in the specified frequency channel. This command is used for Scanning function. User send different frequency value by this command and wait for the reply of the module, then can find if matched signal exist and decide whether need to turn to next channel.

Format: S+Rx\_F

Reply Format: S=X

0—> Signal matched on the frequency channel, 1—> no signal found

example:

S+455.2250

S=0

### 3.4) Volume Configuration

Format: AT+DMOSETVOLUME=X

Reply: +DMOSETVOLUME: 0

“X” is the volume level, the range is 1---8.

Example: AT+DMOSETVOLUME=1

### 3.5) Audio Filet Command

Format: AT+SETFILTER=PRE/DE-EMPH, HIGHPASS, LOWPASS

Reply: +DMOSETFILTER: X

“X”:

0: command succeed,

1: command failed

PRE/DE-EMPH:

1: emphasis bypass

0: emphasis normal

HIGHPASS:

1: voice\_highpass\_filter\_bypass

0: voice\_highpass\_filter normal

LOWPASS:

1: voice\_lowpass\_filter\_bypass

0: voice\_lowpass\_filter normal

For example:

AT+SETFILTER=0, 0, 0

+DMOSETFILTER: 0

Schedule 1: CDCSS coding

	Transform Bit Pattern	Hex Bit
--	-----------------------	---------

Code		Pattern
023I	11001000000111000110111	640E37
025I	10101000000111101101011	540F6B
026I	01101000000110111010011	340DD3
031I	10011000000111111000101	4C0FC5
032I	01011000000110101111101	2C0D7D
043I	11000100000101101101101	620B6D
047I	11100100000110111111000	720DF8
051I	10010100000101010011111	4A0A9F
054I	00110100000100101111011	1A097B
065I	10101100000110001011101	560C5D
071I	10011100000110011110011	4E0CF3

<b>072I</b>	01011100000111001001011	2E0E4B
<b>073I</b>	11011100000101100111010	6E0B3A
<b>074I</b>	00111100000111100010111	1E0F17
<b>114I</b>	00110010000101111010110	190BD6
<b>115I</b>	10110010000111010100111	590EA7
<b>116I</b>	01110010000110000011111	390C1F
<b>125I</b>	10101010000111011110000	550EF0
<b>131I</b>	10011010000111001011110	4D0E5E
<b>132I</b>	01011010000110011100110	2D0CE6
<b>134I</b>	00111010000110110111010	1D0DBA
<b>143I</b>	11000110000101011110110	630AF6
<b>152I</b>	01010110000100110111100	2B09BC



<b>155I</b>	10110110000110110010001	5B0D91
<b>156I</b>	01110110000111100101001	3B0F29
<b>162I</b>	01001110000100111101011	2709EB
<b>165I</b>	10101110000110111000110	570DC6
<b>172I</b>	01011110000111111010000	2F0FD0
<b>174I</b>	00111110000111010001100	1F0E8C
<b>205I</b>	10100001000110010111011	508CBB
<b>223I</b>	11001001000101110001011	648B8B
<b>226I</b>	01101001000100001101111	34886F
<b>243I</b>	11000101000111011010001	628ED1
<b>244I</b>	00100101000101011111100	128AFC
<b>245I</b>	10100101000111110001101	528F8D

<b>251I</b>	10010101000111100100011	4A8F23
<b>261I</b>	10001101000111101110100	468F74
<b>263I</b>	11001101000100010111101	6688BD
<b>265I</b>	10101101000100111100001	5689E1
<b>271I</b>	10011101000100101001111	4E894F
<b>306I</b>	01100011000111110011000	318F98
<b>311I</b>	10010011000110110001110	498D8E
<b>315I</b>	10110011000101100011011	598B1B
<b>331I</b>	10011011000101111100010	4D8BE2
<b>343I</b>	11000111000111101001010	638F4A
<b>346I</b>	01100111000110010101110	338CAE
<b>351I</b>	10010111000111010111000	4B8EB8

<b>364I</b>	00101111000110100001011	178D0B
<b>365I</b>	10101111000100001111010	57887A
<b>371I</b>	10011111000100011010100	4F88D4
<b>411I</b>	10010000100101101110111	484B77
<b>412I</b>	01010000100100111001111	2849CF
<b>413I</b>	11010000100110010111110	684CBE
<b>423I</b>	11001000100110011101001	644CE9
<b>431I</b>	10011000100110100011011	4C4D1B
<b>432I</b>	01011000100111110100011	2C4FA3
<b>445I</b>	10100100100100011101111	5248EF
<b>464I</b>	00101100100101111110010	164BF2
<b>465I</b>	10101100100111010000011	564E83

<b>466I</b>	01101100100110000111011	364C3B
<b>503I</b>	11000010100101100011110	614B1E
<b>506I</b>	01100010100100011111010	3148FA
<b>516I</b>	01110010100111011000001	394EC1
<b>532I</b>	01011010100111000111000	2D4E38
<b>546I</b>	01100110100101111001100	334BCC
<b>565I</b>	10101110100111100011000	574F18
<b>606I</b>	01100001100110011011101	30CCDD
<b>612I</b>	01010001100110001110011	28CC73
<b>624I</b>	00101001100110101111000	14CD78
<b>627I</b>	11101001100111111000000	74CFC0
<b>631I</b>	10011001100100010100111	4CC8A7

<b>632I</b>	01011001100101000011111	2CCA1F
<b>654I</b>	00110101100111000011001	1ACE19
<b>662I</b>	01001101100111100010010	26CF12
<b>664I</b>	00101101100111001001110	16CE4E
<b>703I</b>	11000011100111010100010	61CEA2
<b>712I</b>	01010011100110111101000	29CDE8
<b>723I</b>	11001011100100011001110	65C8CE
<b>731I</b>	10011011100100100111100	4DC93C
<b>732I</b>	01011011100101110000100	2DCB84
<b>734I</b>	00111011100101011011000	1DCAD8
<b>743I</b>	11000111100110110010100	63CD94
<b>754I</b>	00110111100111110000010	1BCF82

**Appendix: Part of the communication refers to C program.(MCU: PIC1939)**

//=====

RAM DEFINE

```
Const                                unsigned                                char
CMD_HAND[15]={0x41,0x54,0x2B,0x44,0x4D,0x4F,0x43,0x4F,0x4E,0x4E,0x45,0x43,0x54,0x0d,0x0a};
unsigned char CMD_SET[15]={0x41,0x54,0x2b,0x44,0x4d,0x4f,0x53,0x45,0x54,0x47,0x52,0x4f,0x55,0x50,0x3d};
unsigned                                char
CMD_VOLUME[16]={0x41,0x54,0x2B,0x44,0x4D,0x4F,0x53,0x45,0x54,0x56,0x4f,0x4c,0x55,0x4d,0x45,0x3d};
unsigned char tx_buf[50]={0};
unsigned char rx_buf[30]={0};
unsigned char tx_len;
unsigned char rx_len;
unsigned char len_txnow;
unsigned char len_rxnow;
unsigned char status_cnt = 2;
```

```
//=====
```

```
void uart_init()  
// MCU UART Initialization (set to standard format)  
{  
    SPBRGH = 0;  
    SPBRG = 23;  
    TXSTA = 0;  
    RCSTA = 0x90;  
    BAUDCON = 0;  
    TXIE = 0;  
    RCIE = 0;  
}
```

```
//-----
```

```
void check_uart()  
// send handshake instruction to module regularly to detect the module's connection status
```

```
{
unsigned char i;

if(Flag.in_rx == 1)
{
    rx_cnt--;
    if(rx_cnt == 0)
    {
        Flag.in_rx = 0;
        Flag.in_tx = 0;
        Flag.cn_fail = 1;
        LED_CTCS = LED_OFF;
        Flag.poweron = 1;
        fresh_display();
        return;
    }
}

if((Flag.in_tx == 1)|| (Flag.in_rx == 1))    // No interleave sending instruction, to ensure the module properly receiving
```



```
instruction.  
    return;  
send_hand();  
}  
  
//-----  
  
void uart_trans_check(void)  
// judge the transmitting instructions is complete or not  
{  
if((Flag.in_tx == 1)&&(len_txnow > tx_len)&&(TXIF == 1))  
{  
    stop_TX();  
// Send over, close the sending UART function.  
    Flag.in_tx = 0;  
    Flag.in_rx = 1;  
    rx_cnt = 2;  
    len_txnow = 0;  
    len_rxnow = 0;  
    tx_len = 0;
```

```
    clr_tx_buf();  
    // Initializes the related registers and flags  
    start_RX();  
    // receiving function is available  
}  
}  
//-----  
void uart_recv_ack(void)  
{  
    if(Flag.in_rx == 0)  
        return;  
    if(len_rxnow == rx_len)  
    {  
        Flag.in_rx = 0;  
        if((rx_buf[rx_len-3] == 0x30)&&(rx_buf[rx_len-2] == 0x0d)&&(rx_buf[rx_len-1] == 0x0a))  
  
        // judge return instruction  
        {
```

```
status_cnt = 2;
Flag.cn_fail = 0;
LED_CTCS = LED_ON;
fresh_display();
stop_RX();
if((rx_len == 15)&&(Flag.poweron))
{
    Flag.reset = 1;
    Flag.poweron = 0;
}
return;
}
else if(Flag.cn_fail == 1)
    return;
else
{
    status_cnt -= 1;
    if(status_cnt == 0)
```

```
// If the Continuous instruction returns null, the module connection failed flag bit
{
    Flag.cn_fail = 1;
    LED_CTCS = LED_OFF;
    Flag.poweron = 1;
}
}
}
//-----
void send_hand()
{
    unsigned char i;

    for(i=0;i<=14;i++)
        tx_buf[i] = CMD_HAND[i];
    // Load the handshake instruction
    rx_len = 15;
```

```
tx_len = 15;
// Write handshake instruction to send and receive data bytes
len_txnow = 0;
Flag.in_tx = 1;
clr_rx_buf();
// Clear the receive buffer
start_TX();
// send UART function is available
}
//-----
void send_set()
{
unsigned char i;

for(i=0;i<=14;i++)
    tx_buf[i] = CMD_SET[i];
tx_buf[15] = ASCII(Flag.gbw);
tx_buf[16] = ASCII_comma;
```

```
ASCII_TFV();  
tx_buf[25] = ASCII_comma;  
ASCII_RFV();  
tx_buf[34] = ASCII_comma;  
tx_buf[35] = ASCII(Tx_ctcs_3);  
tx_buf[36] = ASCII(Tx_ctcs_2);  
tx_buf[37] = ASCII(Tx_ctcs_1);  
tx_buf[38] = ASCII(Tx_ctcs_0);  
tx_buf[37] = ASCII_comma;  
tx_buf[38] = ASCII(sq);  
tx_buf[39] = ASCII_comma;  
tx_buf[40] = ASCII(Rx_ctcs_3);  
tx_buf[41] = ASCII(Rx_ctcs_2);  
tx_buf[42] = ASCII(Rx_ctcs_1);  
tx_buf[43] = ASCII(Rx_ctcs_0);  
// Instruction of sending data are ASCII  
tx_buf[44] = 0x0d;  
tx_buf[45] = 0x0a;
```

```
// Send instructions all ends with a carriage return line feed (0X0D,0X0A)
rx_len = 16;
tx_len = 46;
// Write handshake instruction to send and receive data bytes
len_txnow = 0;
Flag.in_tx = 1;
clr_rx_buf();
start_TX();
}
//-----
void send_vol()
{
    unsigned char i;

    for(i=0;i<=15;i++)
        tx_buf[i] = CMD_VOLUME[i];
    // load volume instruction
    tx_buf[16] = ASCII(vol);
```

```
tx_buf[17] = 0x0d;
tx_buf[18] = 0x0a;
rx_len = 17;
tx_len = 19;
// write number of bytes to set the volume orders to send and receive data
len_txnow = 0;
Flag.in_tx = 1;
clr_rx_buf();
start_TX();
}
//-----
void clr_tx_buf()
// Clear the send buffer
{
unsigned char i;

for(i=0;i<=39;i++)
    tx_buf[i]=0;
```



```
}  
//-----  
void clr_rx_buf()  
// Clear the receive buffer  
{  
    unsigned char i;  
  
    for(i=0;i<=18;i++)  
        rx_buf[i] = 0;  
}  
//-----  
void start_TX()  
// to make it can send UART  
{  
    TXEN = 1;  
    TXIE = 1;  
}  
//-----
```

```
void stop_TX()
// close UARTsending
{
TXEN = 0;
TXIE = 0;
}
//-----

void start_RX()
// to make it can receive UART
{
CREN = 1;
RCIE = 1;
}
//-----

void stop_RX()
// close UART receiving
{
CREN = 0;
```

```
RCIE = 0;
}
//-----
void interrupt ISR_timer(void)
// interrupt handling
{
    unsigned char int_temp;

    if(TXIF)
    {
        if(Flag.in_tx == 0)
            stop_TX();
        // Not in delivery status, interrupted by mistake
        else if(len_txnow <= tx_len)
        {
            TXREG = tx_buf[len_txnow];
        }
        // update sending data
        len_txnow ++;
```

```
    }  
    else  
        TXIE = 0;  
    // send over  
}  
  
if(RCIF)  
{  
    NOP();  
    if(Flag.in_rx)  
    {  
        rx_buf[len_rxnow] = RCREG;  
        // write the returned dato to receive buffer  
        if((len_rxnow++) == (rx_len+1))  
            stop_RX();  
    }  
    else  
        // Not in receiving state, interrupted by mistake, invalid return values
```

```
{  
    stop_RX();  
    int_temp = RCREG;  
}  
}  
}
```