UART Driver User Guide

V1.00.01

UART Driver

The ISD91XX includes a Universal Asynchronous Receiver/Transmitter (UART). The UART supports high speed operation and flow control functions as well as protocols for Serial Infrared (IrDA) and Local interconnect Network (LIN).

1.1. UART Introduction

The Universal Asynchronous Receiver/Transmitter (UART) performs a serial-to-parallel conversion on data received from the peripheral, and a parallel-to-serial conversion on data transmitted from the CPU. The UART controller also supports LIN (Local Interconnect Network) master mode function and IrDA SIR (Serial Infrared) function. The UART channel supports seven types of interrupts including transmitter FIFO empty interrupt (THRE_INT), receiver threshold level interrupt (RDA_INT), line status interrupt (overrun error or parity error or framing error or break interrupt) (RLS_INT), time out interrupt (TOUT_INT), MODEM status interrupt (MODEM_INT), Buffer error interrupt (BUF_ERR_INT) and LIN receiver break field detected interrupt.
Details please refer to the section in the target chip specification titled UART Interface Controller.

1.2. UART Feature

The UART includes following features:

- 8 bytes entry FIFOs for received and transmitted data payload s
- Auto flow control/flow control function (CTS, RTS) are supported.
- Fully programmable serial-interface characteristics:

  -- 5-, 6-, 7-, or 8-bit character

  -- Even, odd, or no -parity bit generation and detection

  -- 1-, 1&1/2, or 2-stop bit generation

  -- Baud rate generation

-- False start bit detection.

- Support IrDA SIR Function
- Support LIN (Local interconnect network) master mode.
- Programmable baud-rate generator that allows the clock to be divided by programmable divider

1.3. Constant Definition

Constant Name	Value	Description
MODE_TX	0	IRDA or LIN function transmit mode
MODE_RX	1	IRDA or LIN function Receive mode

Constant Name	Value	Description
DRVUART_RDAINT	0x1	Receive Data Available Interrupt and Time-out Interrupt
DRVUART_THREINT	0x2	Transmit Holding Register Empty Interrupt
DRVUART_WAKEUPINT	0x40	Wake up interrupt enable
DRVUART_RLSINT	0x4	Receive Line Interrupt
DRVUART_MOSINT	0x8	MODEM Interrupt
DRVUART_TOUTINT	0x10	Time-out Interrupt
DRVUART_BUFERRINT	0x20	Buffer Error Interrupt Enable
DRVUART_LININT	0x100	LIN RX Break Field Detected Interrupt Enable

Constant Name	Value	Description
DRVUART_DATABITS_5	0x0	Word length select: Character length is 5 bits.
DRVUART_DATABITS_6	0x1	Word length select: Character length is 6 bits.
DRVUART_DATABITS_7	0x2	Word length select: Character length is 7 bits.
DRVUART_DATABITS_8	0x3	Word length select: Character length is 8 bits.

Constant Name	Value	Description
DRVUART_PARITY_NONE	0x0	None parity
DRVUART_PARITY_ODD	0x1	Odd parity enable
DRVUART_PARITY_EVEN	0x3	Even parity enable
DRVUART_PARITY_MARK	0x5	Parity mask
DRVUART_PARITY_SPACE	0x7	Parity space

Constant Name	Value	Description
DRVUART_STOPBITS_1	0x0	Number of stop bit: Stop bit length is 1 bit.
DRVUART_STOPBITS_1_5	0x4	Number of stop bit: Stop bit length is 1.5 bit when character length is 5 bits.
DRVUART_STOPBITS_2	0x4	Number of stop bit: Stop bit length is 2 bit when character length is 6, 7 or 8 bits.

Constant Name	Value	Description
DRVUART_FIFO_1BYTES	0x0	RX FIFO interrupt trigger level is 1 byte
DRVUART_FIFO_4BYTES	0x1	RX FIFO interrupt trigger level is 4 bytes
DRVUART_FIFO_8BYTES	0x2	RX FIFO interrupt trigger level is 8 bytes
DRVUART_FIFO_14BYTES	0x3	RX FIFO interrupt trigger level is 14 bytes
DRVUART_FIFO_30BYTES	0x4	RX FIFO interrupt trigger level is 30 bytes
DRVUART_FIFO_46BYTES	0x5	RX FIFO interrupt trigger level is 46 bytes
DRVUART_FIFO_62BYTES	0x6	RX FIFO interrupt trigger level is 62 bytes

1.4. Type Definition

UART_PORT

Enumeration identifier	Value	Description
UART_PORT0	0x000	UART port 0

1.5. Functions

DrvUART_Open

Prototype

```
int32_t
DrvUART_Open (
   UART_PORT u16Port,
   UART_T *sParam
);
```

Description

The function is used to initialize UART. It consists of baud-rate, parity, data-bits, stop-bits, rx-trigger-level and timeout interval settings.

Parameter

u16Port [in]

Specify UART_PORT0.

sParam [in]

Specify the property of UART. It includes

u32BaudRate: Baud rate (Hz)

u8cParity: NONE/EVEN/ODD parity

It could be

DRVUART_PARITY_ NONE (None parity).

DRVUART_PARITY_EVEN (Even parity)

DRVUART_PARITY_ ODD (Odd parity).

u8cDataBits: data bit setting

It could be

DRVUART_DATA_BITS_5 (5 data bits).

DRVUART_DATA_BITS_6 (6 data bits)

DRVUART_DATA_BITS_7 (7 data bits).

DRVUART_DATA_BITS_8 (8 data bits).

u8cStopBits: stop bits setting

It could be

DRVUART_STOPBITS_1 (1 stop bit).

DRVUART_STOPBITS_1_5 (1.5 stop bit)

DRVUART_STOPBITS_2 (2 stop bits).

u8cRxTriggerLevel: Rx FIFO interrupt trigger Level

LEVEL_X_BYTE means the trigger level of UART channel is X bytes. It could be

DRVUART_FIFO_1BYTE, DRVUART_FIFO_4BYTES

DRVUART_FIFO_8BYTES, DRVUART_FIFO_1 4BYTES

DRVUART_FIFO_30BYTES, DRVUART_FIFO_ 46BYTES

DRVUART_FIFO_62BYTES

In UART0 , it could be LEVEL_ 1_BYTE to LEVEL_62_BYTES.

Others, it could be LEVEL_1_BYTE to LEVEL_14_BYTES.

u8TimeOut: Time out value. It represents N-clock cycle and the counting clock is baud rate.

Include

Driver/ DrvUART.h

Return Value

E_SUCCESS: Success.

E_DRVUART_ERR_PORT_INVALID: Wrong UART port configure

E_DRVUART_ERR_PARITY_INVALID: Wrong party setting

E_DRVUART_ERR_DATA_BITS_INVALID: Wrong Data bit setting

E_DRVUART_ERR_STOP_BITS_INVALID: Wrong Stop bit setting

E_DRVUART_ERR_TRIGGERLEVEL_INVALID: Wrong trigger level setting

Example

```
/* Set UART0 under 115200bps, 8 data bits,1 stop bit and none parity and 1
byte Rx trigger level settings. */
STR_UART_T sParam;
sParam.u32BaudRate        = 115200;
sParam.u8cDataBits        = DRVUART_DATABITS_8;
sParam.u8cStopBits        = DRVUART_STOPBITS_1;
sParam.u8cParity          = DRVUART_PARITY_NONE;
sParam.u8cRxTriggerLevel  = DRVUART_FIFO_1BYTES;
DrvUART_Open (UART_PORT0, &sParam);
```

DrvUART_Close

Prototype

```
void DrvUART_Close (
   E_UART_PORT   u16Port
);
```

Description

The function is used to disable UART clock, disable ISR and clear callback function pointer after checking the TX empty.

Parameter

u16Port [in]

Specify UART_PORT0.

Include

Driver/ DrvUART.h

Return Value

None

Example

```
/* Close UART channel 0 */
DrvUART_Close (UART_PORT0);
```

DrvUART_EnableInt

Prototype

```
int32_t DrvUART_ EnableInt (
   UART_PORT     u16Port,
   uint32_t      u32InterruptFlag,
   PFN_DRVUART_CALLBACK pfncallback
);
```

Description

The function is used to enable specified UART interrupt, install the callback function and enable NVIC UART IRQ.

Parameter

u16Port [in]

Specify UART_PORT0.

u32InterruptFlag [in]

DRVUART_LININT: LIN RX Break Field Detected Interrupt Enable

DRVUART_BUFERRINT: Buffer Error Interrupt Enable

DRVUART_ WAKEUPINT: Wakeup Interrupt.

DRVUART_MOSINT: MODEM Status Interrupt.

DRVUART_RLSNT: Receive Line Status Interrupt.

DRVUART_THREINT: Transmit Holding Register Empty Interrupt.

DRVUART_RDAINT: Receive Data Available Interrupt and Time-out Interrupt

DRVUART_TOUTINT: Time-out Interrupt.

pfncallback [in]

Call back function pointer

Include

Driver/ DrvUART.h

Return Value

E_DRVUART_ARGUMENT: Error Parameter.

E_SUCCESS: Success

Note

Use "|"¨ to connect the interrupt flags to enable multiple interrupts simultaneously.

If you call the function twice in a project, the settings are depending on the second setting.

Example

/* Enable UART channel 0 RDA and THRE interrupt. Finally, install UART_INT_HANDLE

function to be callback function. */

DrvUART_EnableInt(UART_PORT0, (DRVUART_RDAINT |

DRVUART_THREINT),UART_INT_HANDLE);

DrvUART_DisableInt

Prototype

```
void   DrvUART_ DisableInt (
   UART_PORT u16Port ,
   uint32_t      u32InterruptFlag
);
```

Description

The function is used to disable UART specified interrupt, uninstall the call back function and disable NVIC UART IRQ.

Parameter

u16Port [in]

Specify UART_PORT0

u32InterruptFlag [in]

DRVUART_LININT: LIN RX Break Field Detected Interrupt Enable

DRVUART_BUFERRINT: Buffer Error Interrupt Enable

DRVUART_WAKEINT: Wakeup Interrupt.

DRVUART_MOSINT: MODEM Status Interrupt.

DRVUART_RLSNT: Receive Line Status Interrupt.

DRVUART_THREINT: Transmit Holding Register Empty Interrupt.

DRVUART_RDAINT: Receive Data Availab le Interrupt and Time-out Interrupt

DRVUART_TOUTINT: Time-out Interrupt.

Include

Driver/ DrvUART.h

Return Value

None

Note

Use “|” to connect the interrupt flags to disable multiple interrupts simultaneously.

Example

/* To disable the THRE interrupt enable flag. */

DrvUART_DisableInt (UART_PORT0, DRVUART_THREINT);

DrvUART_ClearInt

Prototype

```
uint32_t
DrvUART_ClearInt (
   UART_PORT u16Port,
   uint32_t     u32InterruptFlag
);
```

Description

The function is used to clear UART specified interrupt flag.

Parameter

u16Port [in]

Specify UART_PORT0.

u32InterruptFlag [in]

DRVUART_MOSINT: MODEM Status Interrupt.

DRVUART_RLSNT: Receive Line Status Interrupt.

DRVUART_RDAINT: Receive Data Available Interrupt.

DRVUART_TOUTINT: Time-out Interrupt.

DRVUART_THREINT: Transmit Holding Register Empty Interrupt.

Include

Driver/ DrvUART.h

Return Value

E_SUCESS	Success

Example

```
/* To clear UART0 Receive Line interrupt flag */
DrvUART_ClearInt (UART_PORT0, DRVUART_RLSNT);
```

DrvUART_GetIntStatus

Prototype

```
int8_t
DrvUART_GetIntStatus (
   UART_PORT u16Port,
   uint32_t   u32InterruptFlag
);
```

Description

The function is used to get the specified UART interrupt status.

Parameter

u16Port [in]

Specify UART_PORT0

u32InterruptFlag [in]

DRVUART_LININT: LIN RX Break Field Detected Interrupt Enable

DRVUART_BUFERRINT: Buffer Error Interrupt Enable

DRVUART_WAKEINT: Wakeup Interrupt.

DRVUART_MOSINT: MODEM Status Interrupt.

DRVUART_RLSNT: Receive Line Status Interrupt.

DRVUART_THREINT: Transmit Holding Register Empty Interrupt.

DRVUART_RDAINT: Receive Data Availab le Interrupt.

DRVUART_TOUTINT: Time-out Interrupt.

Include

Driver/ DrvUART.h

Return Value

0: The specified interrupt did not happen.

1: The specified interrupt happened.

E_DRVUART_ARGUMENT: Error Parameter.

Note

It is recommended to poll one interrupt at a time.

Example

```
/* To get the THRE interrupt enable flag. */
If(DrvUART_GetIntStatus (UART_PORT0, DRVUART_THREINT))
   printf("THRE INT is happened!\n");
else
   printf("THRE INT is not happened or error parameter\n");
```

DrvUART_GetCTS

Prototype

```
void
DrvUART_GetCTS (
   UART_PORT u16Port,
   uint8_t      *pu8CTSValue,
  uint8_t      *pu8CTSChangeState
)
```

Description

The function is used to get CTS pin value and detect CTS change state

Parameter

u16Port [in]

Specify UART_PORT0

pu8CTSValue [in]

Specify the buffer to receive the CTS value. Return current CTS pin state.

pu8CTSChangeState [in]

Specify the buffer to receive the CTS change state. Return CTS pin state is changed or not. 1 means changed and 0 means not yet.

Include

Driver/ DrvUART.h

Return Value

None

Example

```
/* To get CTS pin status and save to u8CTS_value. To get detect CTS change flag and save to u8CTS_state. */
uint8_t u8CTS_value, u8CTS_state;
DrvUART_GetCTS(UART_PORT0,& u8CTS_value,& u8CTS_state);
```

DrvUART_SetRTS

Prototype

```
void
DrvUART_SetRTS (
  UART_PORT u16Port,
  uint8_t      u8Value
)
```

Description

The function is used to set RTS setting.

Parameter

u16Port [in]

Specify UART_PORT0.

u8Value [in]

Set 0: Drive RTS pin to logic 1 (If the LEV_RTS set to low level triggered).

Drive RTS pin to logic 0 (If the LEV_RTS set to high level triggered).

Set 1: Drive RTS pin to logic 0 (If the LEV_RTS set to lo w level triggered).

Drive RTS pin to logic 1 (If the LEV_RTS set to high level triggered).

Note. LEV_RTS is RTS Trigger Level. 0 is low level and 1 is high level.

Include

Driver/ DrvUART.h

Return Value

None

Example

```
/* Condition: Drive RTS to logic 1 in UART channel 0 and Set RTS trigger
level is 1 bytes*/
DrvUART_SetRTS (UART_PORT0,1);
```

DrvUART_Read

Prototype

```
int32_t
DrvUART_Read (
   UART_PORT     u16Port
```

```
  uint8_t     *pu8RxBuf,
  uint32_t     u32ReadBytes
);
```

Description

The function is used to read Rx data from RX FIFO and the data will be stored in pu8RxBuf.

Parameter

u16Port [in]

Specify UART_PORT0.

pu8RxBuf [in]

Specify the buffer to receive the data of receive FIFO.

u32ReadBytes [in]

Specify the read bytes number of data.

Include

Driver/ DrvUART.h

Return Value

E_SUCCESS: Success.

E_DRVUART_TIMEOUT: FIFO polling timeout.

Example

```
/* Condition: Read RX FIFO 1 byte and store in bInChar buffer. */
uint8_t bInChar[1];
DrvUART_Read(UART_PORT0 ,bInChar,1);
```

DrvUART_Write

Prototype

```
int32_t
DrvUART_Write(
  UART_PORT   u16Port
  uint8_t     *pu8TxBuf,
  uint32_t     u32WriteBytes
);
```

Description

The function is to write data into TX buffer to transmit data by UART

Parameter

u16Port [in]

Specify UART_PORT0

pu8TxBuf [in]

Specify the buffer to send the data to UART transmission FIFO.

u32WriteBytes [in]

Specify the byte number of data.

Include

Driver/ DrvUART.h

Return Value

E_SUCCESS: Success

E_DRVUART_TIMEOUT: FIFO polling timeout

Example

```
/* Condition: Send 1 byte from bInChar buffer to TX FIFO. */
uint8_t bInChar[1] = 0x55;
DrvUART_Write(UART_PORT0,bInChar,1);
```

DrvUART_SetPDMA

Prototype

```
void
DrvUART_SetPDMA (
   UART_PORT u16Port,
    uint16_t u16IsEnable
   );
```

Description

The function is used to control enable/disable PDMA channel

Parameter

u16Port [in]

Specify UART_PORT0

u16IsEnable[in]

Enable TX/RX PDMA TRUE or FASLE.

Include

Driver/ DrvUART.h

Return Value

None.

Example

/* Enable TX and RX PDMA in UART 0 */

DrvUART_EnablePDMA(UART_PORT0, 1);

DrvUART_BaudRateCalculator

Prototype

```
void
DrvUART_BaudRateCalculator (
    uint32_t i32clk,
    uint32_t i32baudRate,
    UART_BAUD_T *baud
    );
```

Description

The function is used to get compute Baud Setting Value.

Parameter

i32clk [in]

Uart Source Clock; unit: Hz

i32baudRate [in]

User seting BaudRate; unit: Bits per second.

computer: 110; 300; 1200; 2400; 4800; 9600; 19200; 38400; 57600; 115200; 230400; 460800; 921600

baud [in]

Get User Settings.

Include

Driver/ DrvUART.h

Return Value

None.

Example

```
/* UART baudrate setting: 115200bps */
DrvUART_BaudRateCalculator(50000000,15200,
&UART0->BAUD,&UART0->BAUD);
```

DrvUART_Init

Prototype

```
void
DrvUART_Init (
   int baudrate
   );
```

Description

The function is used to initialize the UART settings.

Parameter

baudrate [in]

User sets baudrate; unit: Bits per second.

Include

Driver/ DrvUART.h

Return Value

None.

Example

```
  /*Set UART baudrate: 115200bps */
DrvUART_Init(115200);
```

DrvUART_IsIntEnabled

Prototype

```
uint32_t
DrvUART_IsIntEnabled(
  UART_PORT u16Port,
  uint32_t u32InterruptFlag
 );
```

Description

The function is used to get the interrupt enable status.

Parameter

u16Port [in]

Specify UART_PORT0.

u32InterruptFlag [in]

DRVUART_LININT/DRVUART_WAKEUPINT/
DRVUART_BUFERRINT/DRVUART_RLSNT/
DRVUART_MOSINT/DRVUART_THREINT/DRVUART_RDA
INT/DRVUART_TOUTINT.

Include

Driver/ DrvUART.h

Return Value

Specified Interrupt Flag Set or clear.

Example

```
/* check if "wake up CPU function" is enable or not in UART 0 */
DrvUART_IsIntEnabled (UART_PORT0, DRVUART_WAKEUPINT);
```

DrvUART_kbhit

Prototype

```
int32_t
DrvUART_kbhit (
 void
 );
```

Description

This function returns TRUE when UART get any character. Default use UART0.

Parameter

None

Include

Driver/ DrvUART.h

Return Value

1: UART get any character

0: UART does not get any character

Example

```
  /* Check if UART 0 get any character. */
int32_t    uart_w;
uart_w = DrvUART_kbhit();
```

DrvUART_OpenIRCR

Prototype

```
void
DrvUART_OpenIRCR(
   UART_PORT u16Port,
   STR_IRCR_T str_IRCR )
  );
```

Description

The function is to Set IRCR Control Register.

Parameter

u16Port [in]

Specify UART_PORT0.

str_IRCR [in]

The structure of IRCR

It includes of

u8cTXSelect : 1 : Enable IRCR transmit function. It becomes TX mode.

0 : Disable IRCR transmit function.

u8cRXSelect : 1 : Enable IRCR receive function. It becomes RX mode.

0 : Disable IRCR receive function.

u8cInvTX : Invert Tx signal.

u8cInvRX : Invert Rx signal.

Include

Driver/ DrvUART.h

Return Value

None.

Note

Befo re using the API, you should configure UART setting firstly. And make sure the baud-rate setting is used mode 0 (UART divider is 16) in baud-rate configure.

Example

```
/* Change UART0 to IRCR function and Inverse the RX signals. */
STR_IRCR_T sIrda;
sIrda.u8cTXSelect      = ENABLE;
sIrda.u8cInvTX          = FALSE;
sIrda.u8cInvRX          = TRUE;
DrvUART_OpenIRCR (UART_PORT0,&sIrda);
```

DrvUART_OpenLIN

Prototype

```
void
DrvUART_OpenLIN (
   UART_PORT u16Port,
    uint16_t u16DIRECTION,
    uint16_t u16BCNT
   );
```

Description

The function is used to set LIN relative setting.

Parameter

u16Port [in]

Specify UART_PORT0.

u16DIRECTION [in]

MODE_TX or MODE_RX.

u16BCNT [in]

Break Count.

Include

Driver/ DrvUART.h

Return Value

E_SUCCESS success.

Example

```
/* Change UART0 to LIN function and set to transmit the header information.*/
DrvUART_OpenLIN(UART_PORT0,MODE_TX|MODE_RX,13);
```

DrvUART_SetFIFOTriggerLevel

Prototype

```
void
DrvUART_SetFIFOTriggerLevel (
   UART_PORT u16Port,
    uint16_t u16TriggerLevel
   );
```

Description

The function is used to set Rx FIFO Trigger Level.

Parameter

u16Port [in]

Specify UART_PORT0.

U16TriggerLevel [in]

FIFO Trigger Level :LEVEL_1_BYTE to LEVEL_62_BYTES.

Include

Driver/ DrvUART.h

Return Value

None.

Example

```
/* Enable 1 byte trigger level in UART 0 */
DrvUART_SetFIFOTriggerLevel(UART_PORT0, LEVEL_1_BYTE);
```

DrvUART_SetRxTimeOut

Prototype

```
void
DrvUART_SetRxTimeOut (
   UART_PORT u16Port,
   uint8_t u8TimeOut
   );
```

Description

The function is used to set Rx Time Out Value.

Parameter

u16Port [in]

Specify UART_PORT0.

u8TimeOut [in]

Time out value.

Include

Driver/ DrvUART.h

Return Value

None.

Example

```
/* Set Rx timeout = 5 x baud-rate in UART 0 */
DrvUART_SetRxTimeOut(UART_PORT0,5);
```

DrvUART_GetUartCLk

Prototype

```
uint32_t
DrvUART_GetUartCLk (
   void );
```

Description

The function is used to get Uart clock.

Parameter

None

Include

Driver/ DrvUART.h

Return Value

Current Uart Clock.

Example

```
/* Get Uart clock */
uint32_t    uart_clk;
uart_clk    = DrvUART_GetUartCLk ();
```

DrvUART_GetVersion

Prototype

```
int32_t
DrvUART_GetVersion (void);
```

Description

Return the current version number of driver.

Include

Driver/ DrvUART.h

Return Value

Version number:

31:24	23:16	15:8	7:0
00000000	MAJOR_NUM	MINOR_NUM	BUILD_NUM

UART02_IRQHandler

Prototype

void UART02_IRQHandler(void)

Description

Install ISR to handle interrupt event.

Parameter

None

Include

Driver/ DrvUART.h

Return Value

None.

2. Revision History

Version	Date	Description
1.00.01	Mar. 2011	Preliminary UART Driver User Guide of ISD9160