Use ACMP to Invert UART TX

Example Code Introduction for 32-bit NuMicro® Family

Information

Application	Use ACMP to invert the UART TX signal.
BSP Version	NUC230_240 Series BSP CMSIS V3.01.001
Hardware	NuTiny-EVB-NUC240 V1.2

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1 Function Description

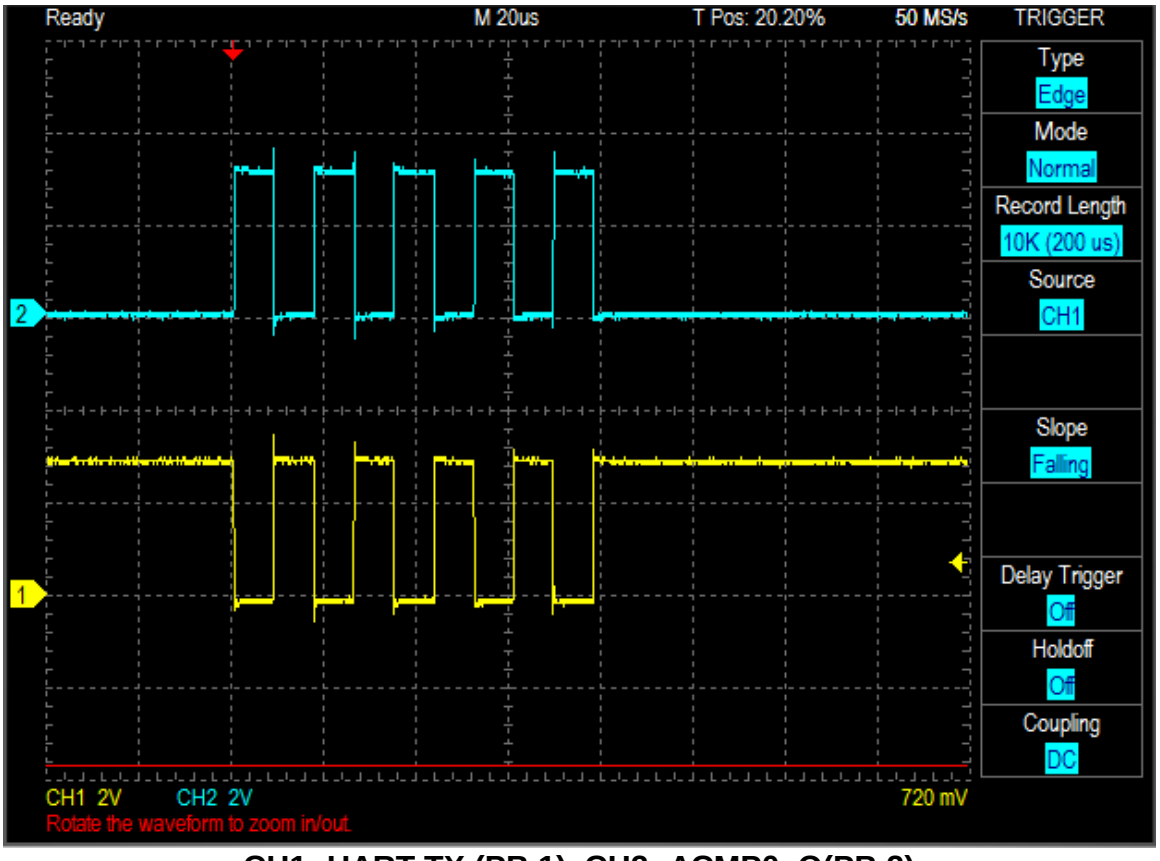
1.1 Introduction

In some applications, the UART signal needs to be inverted, and we can implement it using ACMP. This article uses UART TX as an example, and UART can also use this method.

1.2 Principle

The ACMP inside the MCU is used as an inverter. The TX signal is input to the ACMP_P pin, ACMP_N uses the internal band-gap voltage, ACMP_O is set to the inverted signal output, and the inverted UART signal output can be obtained at ACMP_O.
When ACMP_P is high, ACMP_P is greater than band-gap voltage, and the output is set to signal output inversion, so the output is low; When ACMP_P is low, ACMP_P is less than the BAND GAP voltage, and the output is set to the inverted signal output, so the output is high. Therefore, the UART signal can be inverted by ACMP.

1.3 Demo Result



CH1: UART TX (PB.1), CH2: ACMP0_O(PB.2)

2 Code Description

```
/*---------------------------------------------------------------------------*/
/* Define                                                                    */
/*---------------------------------------------------------------------------*/
#define TEST_NUM 0x55

/*---------------------------------------------------------------------------*/
/* Init System Clock                                                         */
/*---------------------------------------------------------------------------*/
void SYS_Init(void)
{

    /* Unlock protected registers */
    SYS_UnlockReg();

    /* Enable Internal RC 22.1184 MHz clock */
    CLK_EnableXtalRC(CLK_PWRCON_OSC22M_EN_Msk);

    /* Waiting for Internal RC clock ready */
    CLK_WaitClockReady(CLK_CLKSTATUS_OSC22M_STB_Msk);

    /* Enable external 12 MHz XTAL */
    CLK_EnableXtalRC(CLK_PWRCON_XTL12M_EN_Msk);

    /* Waiting for clock ready */
    CLK_WaitClockReady(CLK_CLKSTATUS_XTL12M_STB_Msk);

    /* Set core clock rate as 72 MHz from PLL */
    CLK_SetCoreClock(72000000);

    /* Select HXT as the clock source of UART */
    CLK_SetModuleClock(UART0_MODULE, CLK_CLKSEL1_UART_S_HXT, CLK_CLKDIV_UART(1));

    /* Enable UART peripheral clock */
    CLK_EnableModuleClock(UART0_MODULE);
    /* Enable ACMP01 peripheral clock */
    CLK_EnableModuleClock(ACMP_MODULE);
```

```
    /* Update System Core Clock */
    /* User can use SystemCoreClockUpdate() to calculate SystemCoreClock and CyclesPerUs
       automatically. */
    SystemCoreClockUpdate();


   /*---------------------------------------------------------------------*/
   /* Init I/O Multi-function                                             */
   /*---------------------------------------------------------------------*/

    /* Set PC.6 multi-function pin for ACMP0 positive input pin */
    SYS->GPC_MFP = (SYS->GPC_MFP & ~SYS_GPC_MFP_PC6_Msk) | (SYS_GPC_MFP_PC6_ACMP0_P);
    SYS->ALT_MFP1 = (SYS->ALT_MFP1 & ~SYS_ALT_MFP1_PC6_Msk) | (SYS_ALT_MFP1_PC6_ACMP0_P);

    /* Disable digital input path of analog pin PC.6(ACMP0_P) to prevent leakage */
    GPIO_DISABLE_DIGITAL_PATH(PC, (1ul << 6));

    /* Set PB.2 multi-function pin for ACMP0 output pin */
    SYS->ALT_MFP = (SYS->ALT_MFP & ~SYS_ALT_MFP_PB2_Msk) | (SYS_ALT_MFP_PB2_ACMP0_O);
    SYS->ALT_MFP2 = (SYS->ALT_MFP2 & ~SYS_ALT_MFP2_PB2_Msk) | (SYS_ALT_MFP2_PB2_ACMP0_O);
    SYS->GPB_MFP = (SYS->GPB_MFP & ~SYS_GPB_MFP_PB2_Msk) | SYS_GPB_MFP_PB2_ACMP0_O;

    /* Set PB multi-function pins for UART0 RXD and TXD */
    SYS->GPB_MFP = (SYS->GPB_MFP & ~(SYS_GPB_MFP_PB0_Msk | SYS_GPB_MFP_PB1_Msk)) |
    (SYS_GPB_MFP_PB0_UART0_RXD | SYS_GPB_MFP_PB1_UART0_TXD);

    /* Lock protected registers */
    SYS_LockReg();
}

int32_t main(void)
{

    /* Init system, peripheral clock and multi-function I/O */
    SYS_Init();

    /* Configure UART0: 115200, 8-bit word, no parity bit, 1 stop bit. */
    UART_Open(UART0, 115200);

    /* Configure ACMP0 and select internal reference voltage as negative input. */
    /* ACMP0 positive input needs to be connected to UART0_TX pin. */
    ACMP_Open(ACMP, 0, ACMP_CR_VNEG_BANDGAP, ACMP_CR_HYSTERESIS_DISABLE);
```

```
    /* Enable ACMP0 output inverse function. */
    ACMP_ENABLE_OUTPUT_INVERSE(ACMP,0);


    /* Print TEST_NUM to UART TX pin. */
    /* User can use the oscilloscope to measure the UART TX signal(PB.1) and the
       inverted signal(PB.2). */
    UART_WRITE(UART0, TEST_NUM);
    while(1);

}
```

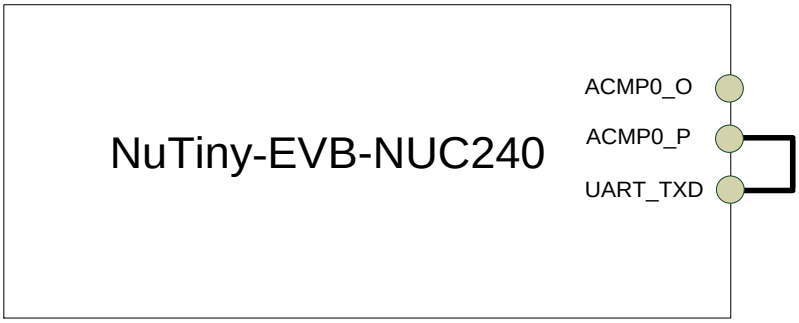
3 Software and Hardware Environment

Software Environment

- BSP version
  - NUC230_240 Series BSP CMSIS V3.01.001
- IDE version
  - Keil uVersion 5.25

Hardware Environment

- Circuit components
  - NuTiny-EVB-NUC240 V1.2
- Diagram



4 Directory Information

- EC_NUC240_ACMP_Invert_UART_TX_V1.00
  - Library — Sample code header and source files
    - CMSIS — Cortex® Microcontroller Software Interface Standard (CMSIS) by Arm® Corp.
    - Device — CMSIS compliant device header file
    - StdDriver — All peripheral driver header and source files
  - SampleCode
    - ExampleCode — Source file of example code

5 How to Execute Example Code

1. This project supports Keil uVersion 5.25 and higher version
2. Browsing into sample code folder by Directory Information (section 4) and double click ACMP_Invert_UART_TX.uvproj
3. Enter the Keil compile mode window
    a. Build the project
    b. Download
    c. Start / Stop the debug mode
4. Enter debug mode
    a. Run

6 Revision History

Date	Revision	Description
Jul. 01, 2019	1.00	1. Initially issued.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.