UART Implementation Using the N2HET

Haixiao Weng *MCU Safety Application*

ABSTRACT

This application report describes how the Hercules N2HET peripheral can be used to implement UART with interrupt capability and zero CPU overhead. The solution applies to all the Hercules MCUs that have N2HET modules such as TMS570 series and RM series devices.

Contents

Figures

Tables

No table of figures entries found.

1 Introduction

The N2HET peripheral is a complex high-performance RISC coprocessor that operates independently from the main Cortex™ R4F CPU and can be used to implement complex timed I/O operations running in the background. This is the enhanced version of the HET in TMS470(M) series and the NHET in TMS570LS20x series. More information on the N2HET can be found in the Hercules TRM and datasheet.

In this application report, the N2HET is used to implement UART interface. The goal is to provide hardware UART-module-like functionality with independent background transmission/reception

The N2HET program runs in the background, independent of the main Cortex™ R4F CPU, and performs all the tasks associated with the UART communication. The incoming data stream delivered to an N2HET device I/O pin is decoded. Upon the reception of a full character, the Cortex™ R4F CPU program is notified, and it then can directly fetch the entire received data byte from the N2HET internal RAM. For outgoing data streams, the Cortex™ R4F CPU just passes the data to be transmitted to the N2HET program and initiates the transmission. After this, the entire transmission is handled by the N2HET program, and a data stream is output to an N2HET device I/O pin. During this process, the Cortex™ R4F CPU is free for application-related tasks. Figure 1 shows a conceptual overview of the interaction among different components. It illustrates the hardware setup for the attached example as well.

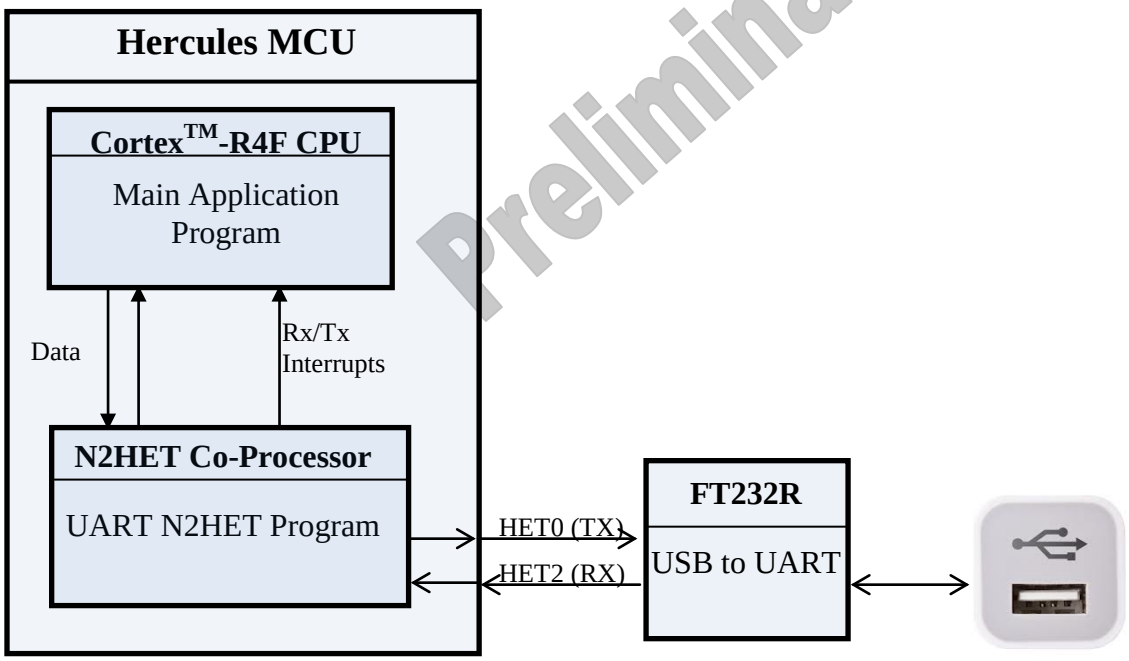

Figure 1. Hercules N2HET Emulated UART Implementation Overview

2 N2HET Emulated UART

In this UART implementation, the N2HET program processes the incoming and outgoing data streams bit by bit. The N2HET UART program must adhere to the general rule that the entire N2HET program must be processed within one loop resolution (LR) clock period. In the N2HET design, the CPU can write to the N2HET RAM while the N2HET is executing from another position. Therefore, the time slots for the Cortex™ R4F CPU to accessing the N2HET RAM can be saved. The N2HET UART code presented in this application report consists of independent entities that implement transmission and reception. These building blocks can be combined to create any number of transmission or reception channels as required by the main application, as long as the N2HET requirements are met.

In the attached example, HCLK = 180MHz, VCLK = 90MHz and the loop resolution (LR) is 90MHz/(4x15) =1.5MHz. The emulate UART baud-rate is 115200 bit/s. This means the N2HET program transmits or receives each bit for duration of 13 loop resolution periods (LRPs). 13 is just an arbitrary number and user can customize this number based on the PLL setting and the UART baud rate tolerance.

The user must pay attention to the available time slots whenever changing the UART baud rate or implement multiple instances. For example, in the current implementation, a maximum of 21 (worst case) N2HET instructions are executed in one LRP and the available time slots is 60, so the maximum instances of 115200 bit/s UART per N2HET is 2 in this case. There are some methods to reduce the number of time slots required for the current implementation:

- When multiple instances UART are implemented in one N2HET, user can force the TX part of different instances shift by one LRP, for example:

```
    SCITX    CNT     { next= Lm10,  reg = T, max= 12, irq = OFF,data=1}
```

  Here, the '1' is an offset from the 1st instance.

- User can use MCMP or ECMP instruction to replace the combo of SUB and BR instructions. However, user needs to assign spare N2HET pins to MCMP or ECMP.

2.1 Data Transmission

A UART transmission is initiated by the Cortex™ R4F program by calling the function HetUART1PutChar(). This function first adds a zero start bit and a one stop bit to the data to be sent. This value then gets loaded into the data field of the SCITX N2HET instruction, which serves as a transmit buffer. Then, the number of bits to be transmitted is loaded into Lm11. In the case of the attached example, this is ten, as the communication scheme used is one start bit, eight data bits, no parity bit, and one stop bit. The parameters that are loaded can easily be adapted to custom requirements to accommodate different data bit lengths or additional stop and parity bits. After the communication data is loaded, the actual transmission which will be performed by the N2HET program. The Cortex™ R4F CPU can continue to process other tasks.

Figure 2 illustrates the N2HET program flow of the UART transmit implementation. The N2HET data transmission routine starts to shift out the data bits after the number of bits is written into Lm11. Since there are 13 time slots (LRPs) for one UART bit, the N2HET program only shift out the bits every 13 LRPs. This is realized by a counter from 0 to 12. The SendOverINT instruction will generate an transmit buffer ready interrupt once all the 10 TX bits (one start, eight data bits and one stop bit) are shifted out.

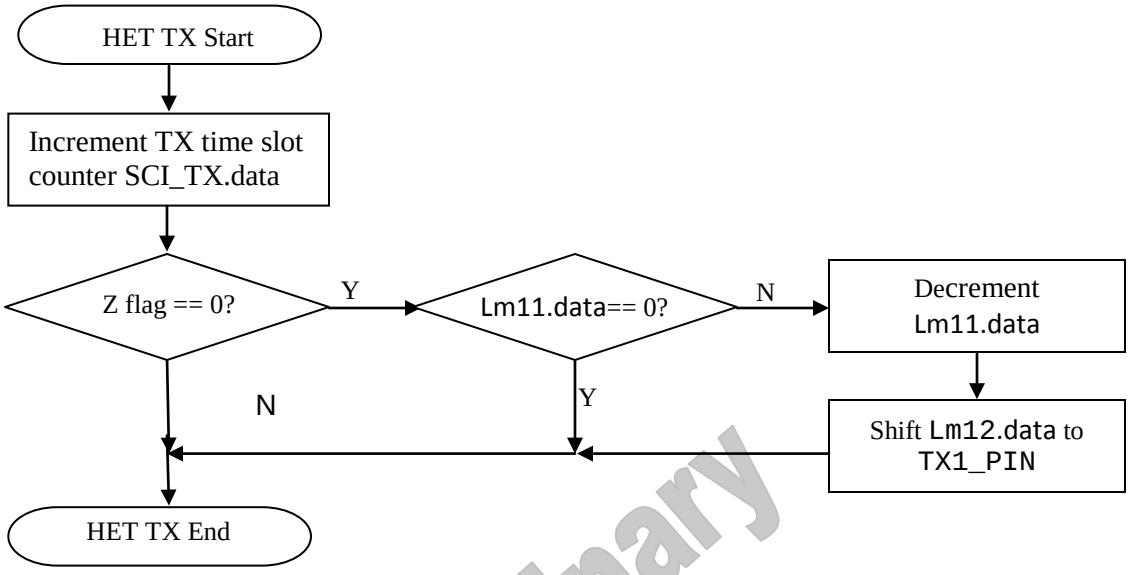

Figure 2. N2HET Emulated UART Transmit Program FlowChart

2.2 Data Reception

Figure 3 shows the UART bit timing used in this application note and Figure 4 illustrates the N2HET program flow of the UART reception implementation. Note that when it is indicated that data is stored at or loaded from an N2HET program location, this always refers to the data field that is embedded in the instruction. The N2HET program constantly polls the receive signal line with a frequency 13 times of the baud rate. Each bit therefore consists of 13 samples (one for each LRP). The N2HET program detects a valid start bit if the first four samples are of logic level 0. As soon as a falling edge is detected on RX1_PIN, the N2HET program assumes that a frame is being received and synchronizes itself to the bus.

To prevent interpreting noise as Start bit UART expects RX1_PIN line to be low for at least four contiguous N2HET LRPs to detect a valid start bit. The bus is considered idle if this condition is not met. When a valid start bit is detected, the N2HET program determines the value of each bit by sampling the RX1_PIN line value during the 5th, 6th and 7th N2HET LRPs. A majority vote of these three samples is used to determine the value stored in the UART receiver shift register. By sampling in the middle of the bit, the N2HET program reduces errors caused by propagation delays and rise and fall times and data line noises.

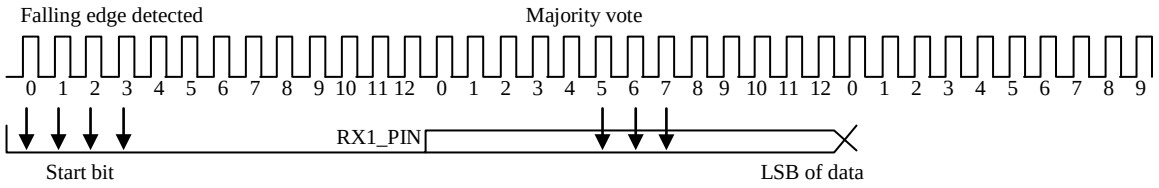

Figure 3. UART Communication Bit Timing

At the end of one bit N2HET time slot, the majority vote RX value will be shift into Bit1RecQ data field. The number of bits received is stored in the data field of counter NumOfBits. This process continues until these counter overflows (reset to 0). Then, the received data is transferred from Bit1RecQ to CopyRem1, and a Cortex™ R4F interrupt is generated by the instruction DoneRec. This also implements a double-buffering scheme, effectively allowing the N2HET program to receive a byte while one is waiting to be read out by the application program. The main program reads out the received value directly from the data field of CopyRem1. This is demonstrated in the function HetUART1GetChar().

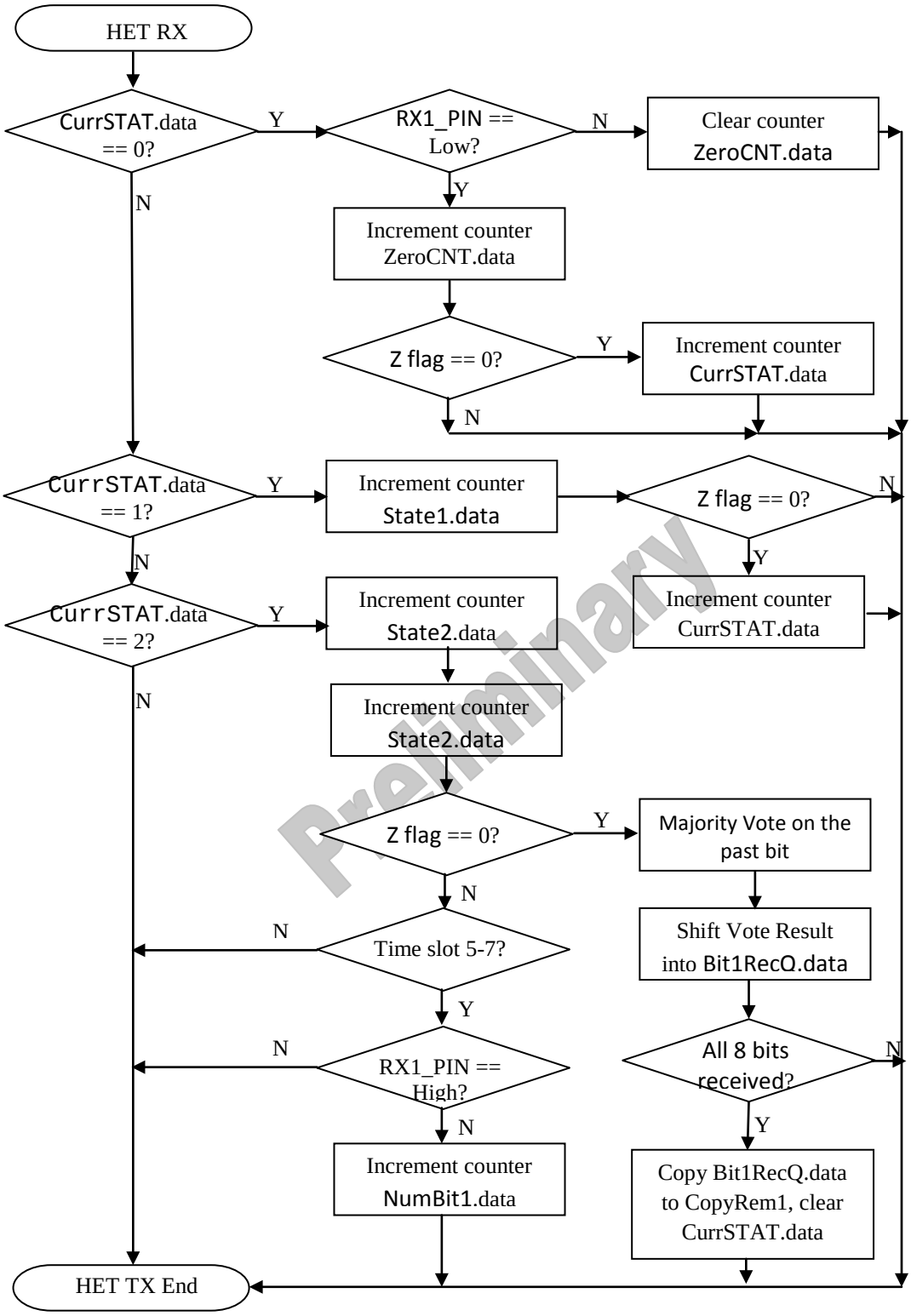

Figure 4. N2HET Emulated UART Receive Program FlowChart

3 Examples

Two example CCS projects are provided to demo the N2HET emulated UART, one works for the TMS570LS series and the other one works for the RM4x series. The N2HET clock is set to 90MHz and the N2HET emulated UART baud rate to 115.2kbit/s, no parity bit, one stop bit. Suppose you follow the hardware setup in Figure 1, import the examples to CCS5.3, download the code to a TMS570LS1227, TMS570LS3137, RM46L852 or RM48L950 device, and run it, the following message will show up on the HyperTerminal:

"Send Data using Polling

BaudRate: 115200 bps

Send Data using Interrupt

Type Any Letter to Echo using Polling:"

After the user type any letter in the HyperTerminal, the letter will echo back. Then, the program will switch to interrupt receive mode:

"Type Any Letter to Echo using Interrupt:"

Any letter typed afterwards will echo back on the HyperTerminal through the receive interrupts.

If you port the examples to a different Hercules device other than those listed in the form paragraph, please check the device datasheet to make sure that your device's HCLK frequency can tolerance 180MHz. Otherwise, please adjust the PLL settings to slow down the HCLK, meanwhile, adjust the HCLK to VCLK2 ratio in CLKCNTL register (@0xFFFFFFD0) and HETPFR register (@0xFFF7B804 for N2HET1 or @0xFFF7B904 for N2HET2) to keep the same LRP – 1.5MHz. If the use case requires a different baud rate, please adjust the LRP accordingly.

Reference

1. *TMS570LS1227 Technical Reference Manual* (SPNU515)
2. *TMS570LS1227 Data Sheet* (SPNS192)
3. *RM46L852 Technical Reference Manual* (SPNU514)
4. *RM46L852 Data Sheet* (SPNS185)