Software that sends and receives data must implement a mechanism to synchronize the software with the hardware. In particular, the software should read data from the input device only when data is indeed ready. Similarly, software should write data to an output device only when the device is ready to accept new data. With busy-wait synchronization, the software continuously checks the hardware status waiting for it to be ready. In this section, we will use busy-wait synchronization to write I/O programs that send and receive data using the UART. After a frame is received, the receive FIFO will be not empty (**RXFE** becomes 0) and the 8-bit data is available to be read. To get new data from the serial port, the software first waits for **RXFE** to be zero, then reads the result from **`UART1_DR_R`**. Recall that when the software reads **`UART1_DR_R`** it gets data from the receive FIFO. This operation is illustrated in Figure 11.8 and shown in Program 11.1. In a similar fashion, when the software wishes to output via the serial port, it first waits for **TXFF** to be clear, then performs the output. When the software writes **`UART1_DR_R`** it puts data into the transmit FIFO.

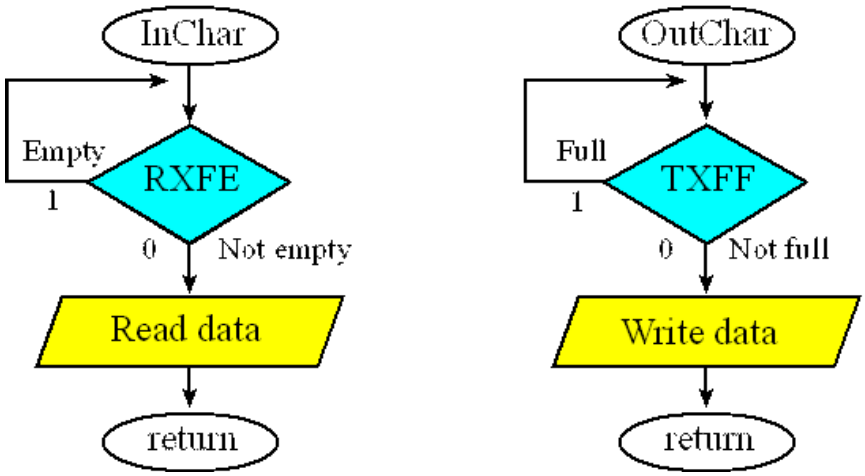

Figure 11.8. Flowcharts of InChar and OutChar using busy-wait synchronization.

The initialization program, **`UART_Init`**, enables the UART1 device and selects the baud rate. The **PCTL** bits were defined back in Chapter 6, and repeated as Table 11.3. **PCTL** bits 5-4 are set to 0x22 to select U1Tx and U1Rx on PC5 and PC4. The input routine waits in a loop until **RXFE** is 0 (FIFO not empty), then reads the data register. The output routine first waits in a loop until **TXFF** is 0 (FIFO not full), then writes data to the data register. Polling before writing data is an efficient way to perform output. **UART2_xxx.zip** is the interrupt-driven version. Be careful when using Port C to be friendly; the pins PC3-PC0 are used by the debugger and you should not modify their configurations.

IO	Ain	0	1	2	3	4	5	6	7	8	9	14
PA0		*Port*	*U0Rx*							CAN1Rx		
PA1		*Port*	*U0Tx*							CAN1Tx		
PA2		Port		SSI0Clk								
PA3		Port		SSI0Fss								
PA4		Port		SSI0Rx								
PA5		Port		SSI0Tx								
PA6		Port			I_2C1SCL		M1PWM2					
PA7		Port			I_2C1SDA		M1PWM3					
PB0		Port	U1Rx						T2CCP0			
PB1		Port	U1Tx						T2CCP1			
PB2		Port			I_2C0SCL				T3CCP0			
PB3		Port			I_2C0SDA				T3CCP1			
PB4	Ain10	Port		SSI2Clk		M0PWM2			T1CCP0	CAN0Rx		
PB5	Ain11	Port		SSI2Fss		M0PWM3			T1CCP1	CAN0Tx		
PB6		Port		SSI2Rx		M0PWM0			T0CCP0			
PB7		Port		SSI2Tx		M0PWM1			T0CCP1			
PC4	C1-	Port	U4Rx	U1Rx		M0PWM6		IDX1	WT0CCP0	U1RTS		
PC5	C1+	Port	U4Tx	U1Tx		M0PWM7		PhA1	WT0CCP1	U1CTS		
PC6	C0+	Port	U3Rx					PhB1	WT1CCP0	USB0epen		
PC7	C0-	Port	U3Tx						WT1CCP1	USB0pflt		
PD0	Ain7	Port	SSI3Clk	SSI1Clk	I_2C3SCL	M0PWM6	M1PWM0		WT2CCP0			
PD1	Ain6	Port	SSI3Fss	SSI1Fss	I_2C3SDA	M0PWM7	M1PWM1		WT2CCP1			
PD2	Ain5	Port	SSI3Rx	SSI1Rx		M0Fault0			WT3CCP0	USB0epen		
PD3	Ain4	Port	SSI3Tx	SSI1Tx				IDX0	WT3CCP1	USB0pflt		
PD4	*USB0DM*	*Port*	*U6Rx*						*WT4CCP0*			
PD5	*USB0DP*	*Port*	*U6Tx*						*WT4CCP1*			
PD6		Port	U2Rx			M0Fault0		PhA0	WT5CCP0			

PD7		Port	U2Tx					PhB0	WT5CCP1	NMI		
PE0	Ain3	Port	U7Rx									
PE1	Ain2	Port	U7Tx									
PE2	Ain1	Port										
PE3	Ain0	Port										
PE4	Ain9	Port	U5Rx		I_2C2SCL	M0PWM4	M1PWM2			CAN0Rx		
PE5	Ain8	Port	U5Tx		I_2C2SDA	M0PWM5	M1PWM3			CAN0Tx		
PF0		Port	U1RTS	SSI1Rx	CAN0Rx		M1PWM4	PhA0	T0CCP0	NMI	C0o	
PF1		Port	U1CTS	SSI1Tx			M1PWM5	PhB0	T0CCP1		C1o	TRD1
PF2		Port		SSI1Clk		M0Fault0	M1PWM6		T1CCP0			TRD0
PF3		Port		SSI1Fss	CAN0Tx		M1PWM7		T1CCP1			TRCLK
PF4		Port					M1Fault0	IDX0	T2CCP0	USB0epen		

Table 11.3. PMCx bits in the GPIOPCTL register on the LM4F/TM4C specify alternate functions. PD4 and PD5 are hardwired to the USB device. PA0 and PA1 are hardwired to the serial port. PWM does not exist on LM4F120.

Help

```
// Assumes a 80 MHz bus clock, creates 115200 baud rate
void UART_Init(void){            // should be called only once
  SYSCTL_RCGC1_R |= 0x00000002;  // activate UART1
  SYSCTL_RCGC2_R |= 0x00000004;  // activate port C
  UART1_CTL_R &= ~0x00000001;    // disable UART
  UART1_IBRD_R = 43;     // IBRD = int(80,000,000/(16*115,200)) = int(43.40278)
  UART1_FBRD_R = 26;     // FBRD = round(0.40278 * 64) = 26
  UART1_LCRH_R = 0x00000070;  // 8 bit, no parity bits, one stop, FIFOs
  UART1_CTL_R |= 0x00000001;     // enable UART
  GPIO_PORTC_AFSEL_R |= 0x30;    // enable alt funct on PC5-4
  GPIO_PORTC_DEN_R |= 0x30;      // configure PC5-4 as UART1
  GPIO_PORTC_PCTL_R = (GPIO_PORTC_PCTL_R&0xFF00FFFF)+0x00220000;
  GPIO_PORTC_AMSEL_R &= ~0x30;   // disable analog on PC5-4
}
// Wait for new input, then return ASCII code
unsigned char UART_InChar(void){
  while((UART1_FR_R&0x0010) != 0);      // wait until RXFE is 0
  return((unsigned char)(UART1_DR_R&0xFF));
}
// Wait for buffer to be not full, then output
void UART_OutChar(unsigned char data){
  while((UART1_FR_R&0x0020) != 0);      // wait until TXFF is 0
```

```
  UART1_DR_R = data;
}
// Immediately return input or 0 if no input
unsigned char UART_InCharNonBlocking(void){
  if((UART1_FR_R&UART_FR_RXFE) == 0){
    return((unsigned char)(UART1_DR_R&0xFF));
  } else{
    return 0;
  }
}
```

Program 11.1. Device driver functions that implement serial I/O (CC11_UART and C11_Network).

CHECKPOINT 11.4

How does the software clear RXFE?

Hide Answer

RXFE is set and cleared by hardware. It means receive FIFO empty. To make it 0 means to put data into the FIFO. Software cannot clear this flag. An incoming UART frame will clear RXFE.

CHECKPOINT 11.5

Help

How does the software clear TXFF?

Hide Answer

TXFF is set and cleared by hardware. It means transmit FIFO full. To make it 0 means to get data from the FIFO. Software cannot clear this flag. An outgoing UART frame will clear TXFF.

CHECKPOINT 11.6

Describe what happens if the receiving computer is operating on a baud rate that is twice as fast as the transmitting computer?

Hide Answer

The data will be received in error (values will not be correct). The receiver could appear to get two input frames for every one frame transmitted. It will probably cause framing errors (FE). It would cause parity errors if active.

CHECKPOINT 11.7

Describe what happens if the transmitting computer is operating on a baud rate that is twice as fast as the receiving computer?

Hide Answer

The data will be received in error (values will not be correct). The receiver will appear to get one input frame for every one frame transmitted. It will probably not cause framing errors (FE). It would cause parity errors if active.

CHECKPOINT 11.8

How do you change Program 11.1 to run at the same baud rate, but the system clock is now 10 MHz.

Hide Answer

10,000,000/115200/16 is 5.4253, which is similar to 5 and 27/64. UART0_IBRD_R is 5 UART0_FBRD_R is 27. The baud rate is 10MHz/(5+27/64)/16 which is 115274 bits/sec.