Using Event Characters with FT8U2XX

Introduction

On FT8U2XX devices, data in the input buffer is released as soon as it becomes full (64 bytes), a status change occurs, or a timer expires. The timer period is 16 msecs and is not configurable. In applications in which short packets of data are received, this can lead to situations where data is held-up in the input buffer and will not be released until the timer expires, resulting in reduced performance. Event characters provide a solution to this problem, and the rest of this note describes their use.

Event Characters

The Windows communications architecture defines event services which allow application software to receive notification that communications events have occurred. For example, an application can be notified that an event character has been detected in the receive data stream.

An event character is a programmable and application specific character whose reception signals an event. More importantly, when event character notification is enabled, the FT8U2XX will empty its input buffer when the event character is received. In other words, reception of the event character makes the received data available immediately, and overrides the 16 msec timer.

Example

Suppose a peripheral, which sends short (less than 64 bytes) ASCII strings terminated by a NULL (0) character, is connected to the serial port. If event character notification is enabled and the event character is set to 0, reception of the NULL character at the end of each string will cause the input buffer to be emptied and the received data to be made available. This is in contrast to the situation where event character notification is not enabled, and the received string and NULL character remain in the input buffer until the 16 msec timer expires.

The following code fragments use Win32 Comm API functions to show how this situation could be set up. Note that, in the interests of simplicity, return codes are ignored and error recovery has been omitted.

```
HANDLE hComm;          // HANDLE of an open port
DCB dcb = {0};         // device control block
DWORD dwCommEvent;
DWORD dwRead;
char ch;

GetCommState(hComm, &dcb);     // get current comms settings

// setup required comms settings - event character only for this example
dcb.EvtChar = 0;

SetCommState(hComm,&dcb);      // set new comms settings

SetCommMask(hComm,EV_RXFLAG); // enable event character notification

WaitCommEvent(hComm,&dwCommEvent,NULL);       // wait on event character

// event character has been received, so call ReadFile() to get data
do {
        if (ReadFile(hComm,&ch,1,&dwRead,NULL)) {
                // process data
        }
        else
                break;
} while (dwRead);
```

Note

The use of event characters described in this note depends on the data having a consistent format. In the example, a NULL character can never be part of an ASCII string except as the string terminator, and so reception of the NULL character can only signify the end of the received data.

It could be that, for some applications, the format of the data may not be appropriate to the use of event character notification. In these cases it may be possible to produce a solution by forcing a status change to occur. For example, suppose an application doesn't use DTR/DSR. These lines could be connected together, and the application could force a status change by sending a command to change DTR. Because DTR has been wired to DSR, changing DTR causes a change on DSR which is seen as a status change by the FT8U2XX. In response to the status change, the FT8U2XX will empty its input buffer.