Application Note

Z8 Encore! XP® 9-Bit UART Implementation

AN014602-1207

Abstract

This Application Note describes how to implement a Universal Asynchronous Receiver/Transmitter (UART) that can communicate in 9-bit mode in a multiprocessor (MP) environment. In this discussion, a number of Z8 Encore! XP® microprocessors (MCU) function as Slaves networked to a single Master that is also a Z8 Encore! XP MCU. This Application Note focuses on setting up the Z8 Encore! XP UART in the 9-bit multiprocessor mode as a host and as a Slave.

Z8 Encore! XP® Flash Microcontrollers

Zilog's Z8 Encore! XP products are based on the new eZ8 CPU and introduce Flash memory to Zilog's extensive line of 8-bit MCU's. Flash memory in-circuit programming capability allows for faster development time and program changes in the field. The high-performance register-to-register based architecture of the eZ8 core maintains backward compatibility with Zilog's popular Z8® MCU.

Z8 Encore! XP MCUs combine a 20 MHz core with Flash memory, linear-register SRAM, and an extensive array of on-chip peripherals. These peripherals make the Z8 Encore! XP suitable for a variety of applications including motor control, security systems, home appliances, personal electronic devices, and sensors.

Discussion

In a typical multidrop network, a host device communicates with a number of Slave devices. Generally, the Physical layer of such a network is based on the RS-485/RS-232/RS-422 standard, and the Data Link layer is often a user-defined protocol. Each Slave is identified by a unique address. Target Slave devices search every byte of a transmitted frame for the address byte that matches their unique respective addresses, resulting in a lot of wasted Slave-CPU processing time.

The new generation of microcontrollers, such as Zilog's Z8 Encore! XP MCU, implement a special feature in their serial communication (UART) protocol to reduce wasted CPU processing time. These serial communication protocols send a transmitting message with a ninth bit that indicates whether the byte that follows is an address byte or a data byte. Thus, a Slave device is able to distinguish an address byte from a data byte according to the ninth bit, which is set by the Master device. The Slave device polls for the ninth bit and, if it is set, matches the address with its own. If a match occurs, the data bytes that follow are received. If the address does not match the Slave device address, the Slave device ignores the remainder of the data bytes, thereby reducing the processing time of the Slave's CPU.

Theory of Operation

In a multiprocessor mode of communication, a number of processors share a common serial bus. In such a setup, the Z8 Encore! XP UART uses the ninth bit for selective communication. In multiprocessor mode, this ninth, or MP bit is transmitted immediately after the 8 bits of data. This MP bit is immediately followed by a stop bit, as displayed in Figure 1.

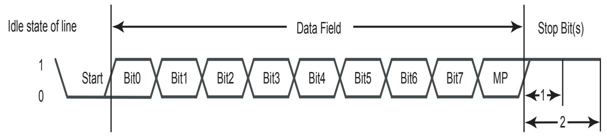

Figure 1. Transmitted Byte

Composition of an Address Byte

Because the Data Link layer of a multidrop network is a user-defined protocol, it offers flexible way of composing a byte, which, in this case, is made up of 9 bits. All the bits contained in an address byte can be used to represent a device address. Conversely, few bits contained in an address byte can be used to represent the address, while the remaining bits can represent a command to the Slave device to perform a predetermined operation.

In this Application Note, three bits of the address byte are used to represent/indicate the address of a Slave device, three bits are used to represent a command, and the remaining two bits are used to represent the length of the data. In this scheme, up to eight Slave devices can be addressed. To increase the number of Slave devices in a network, the number of address bits must be increased as required.

Using three bits to address a device, eight combinations of bits are possible. Of these, seven combinations are used to address seven separate Slave devices, and the remaining combination is used as a common address for all of the Slave devices.

Three bits represent one command, and eight combinations of bits represent eight commands. Of these, some can be used to represent general commands (common to all Slaves) and others can be application/Slave-specific commands. These commands are used to instruct a Slave device to perform a single operation or a set of operations. Some operations are mentioned below:

- Setting the clock to match that of the host, with data being the time in the host device
- Starting/stopping an A/D or D/A conversion
- Setting the I/O lines to a pattern defined by a data word
- Changing the frequency/pulse width of a Pulse Width Modulated (PWM) signal to a value defined by a data byte or bytes

The remaining two bits contained in the address byte are used to represent the number of data bytes following this address byte. The length-of-data bits indicate to the Slave device the number of bytes that it must be ready to receive. This information is useful while setting up the DMA with a UART to relieve the processor of being interrupted for every communication.

Developing the Application with Z8 Encore! XP®

Based on the above discussion about the composition of address bytes, examples showcasing the use of 9-bit UART mode of communication in a network of Z8 Encore! XP devices can be presented.

Consider a networked scenario with one Z8 Encore! XP MCU acting as the host connected to three other Z8 Encore! XP MCU devices acting as Slaves, in multiprocessor mode. Figure 2 displays such a multidrop network.

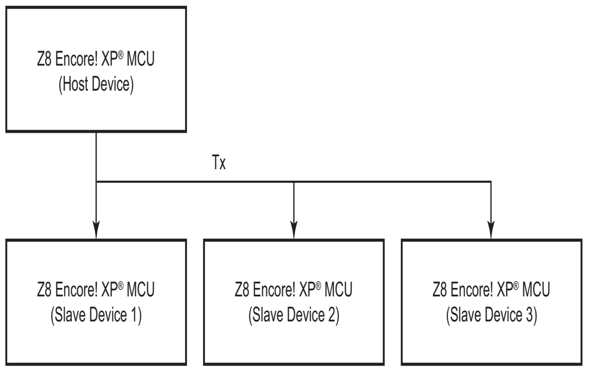

Figure 2. Z8 Encore! XP in a Multidrop Network

The examples that follow in this section highlight how an address byte can be efficiently used to send across a Slave address, a command to the slave, and the length of data that follows this address byte. Figure 3 on page 3 displays such an address byte.

7	6	5	4	3	2	1	0
A	A	A	C	C	C	L	L

Figure 3. Address Byte

AAA—represents a Slave address

CCC—represents a command to the Slave

LL—represents the length of data that follows this address byte

In this Application Note, the address bit combinations are:

000—broadcast message: all the devices read this message

001—corresponds to Device 1

010—corresponds to Device 2

100—corresponds to Device 3

Note: *The address bit combinations 011, 101, 110, and 111 do not represent any device, but can be used to expand the network.*

Table 1 lists commands represented by the 3-bit (C) combinations.

Table 1. Commands and their Bit Combinations

3-Bit Combination	Command	Length of Data to Follow (bytes)
000	Set Frequency for PWM	2
001	Change PWM duty cycle	2
010	Set time (RTC)	4
011	Set / Change port I/O info	1
100	Write I^2C data	4
101	Write to UART1	4
110	Write to RAM Buffer	4
111	Write to FLASH Buffer	4

The length of data that follows the address byte is represented by 2-bit (L) combinations and translate to one, two, three, or four bytes of data.

With this background, consider the following examples.

Example 1

A host system sends a command to all the Slaves in its network to set time to as specified by the four data bytes that follow the address byte. The address byte representation is `0Bh` (00001011) and is displayed in Figure 4.

7	6	5	4	3	2	1	0
A	A	A	C	C	C	L	L
0	0	0	0	1	0	1	1

Figure 4. Address Byte for Example 1

This combination of bits translates to:

- The Slave address is 000 (AAA), indicating a broadcast message that all Slaves in the network are addressed.
- The command is 010 (CCC), indicating that all of the Slaves must set time as time specified by the following bytes of data.

- The length-of-data bits are represented as 11, indicating that 4 bytes of data follow this address byte.

Example 2

A host system sends a command to a specific Slave device, represented by the address bit 001, to change the duty cycle of a PWM signal. In this case, the necessary input is the PWM value represented by the PWM Low byte and PWM High byte. Therefore, the length-of-data bit must represent the value 2. The length of data to be transferred is two bytes. The address byte representation is `26h` (00100110), and is displayed in Figure 5.

7	6	5	4	3	2	1	0
A	A	A	C	C	C	L	L
0	0	1	0	1	1	1	0

Figure 5. Address Byte for Example 2

Software Implementation

The software implementation to illustrate the 9-bit UART mode of communication consists of two modules—the host module and the Slave module. The host module resides in the host system, and sends an address byte to all of the Slave devices in the network using the address bits, the command bits, and the length-of-data bits. The ninth bit of this address byte is set to 1. This address byte is followed by the data bytes, where the ninth bit is set to 0. All the Slave devices are interrupted by the address bytes. Only the addressed Slave device interprets the command and takes appropriate action, using the Slave module residing within it.

In this implementation, the emphasis is on setting up the host for a 9-bit communication, setting up the appropriate control registers, and using the ninth bit to address Slave devices.

The details of the host module and the Slave module are explained in the sections that follow.

Host Module

The host module configures the host device for communicating with Slave devices. It contains the address table and the command table for issuing the addresses and commands to Slave devices. The main function of the host module is to initiate a communication with a Slave device by sending the Slave an address byte followed by a data byte or bytes.

The host module initializes the UART0 (on-chip peripheral) of the Z8 Encore! XP in multiprocessor mode by setting the control register as described in Table 2.

Table 2. UARTx Control 1 Register (UxCTL1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MPM	MPE	MPBT	X	X	X	X	X
Reset	0	0	0	0	0	0	0	0

Note: X = Undefined.

Bit Position	**Value**	**Description**
7	MPM	**Multiprocessor Mode Select** Set this bit to 1 to enable Multiprocessor mode.
6	MPE	**Multiprocessor Enable** Set this bit to 0 to processes all received data bytes.
5	MBPT	**Multiprocessor Bit Transmitter** 1: Send an address to the slave device 0: Send data bytes to the slave device
4:0	—	Reserved.

Further details for setting the control registers are described in the code listings.

Slave Module

The Slave module configures the Slave device to receive communication from the host device and to send communication to the host device and other peripheral devices, depending on the command sent by the host. The Slave device address is hard-coded in the Slave module. To change the Slave address, the `#define` statement in the Slave software (`sio.h` file) must be modified.

The main function of the Slave module is to resolve the address and analyze the command along with the length of data to follow. The Slave device responds only if the address received matches the hard-coded address for the Slave device. Hard-coded addresses are distinct for each Slave device. Separate functions for reading and writing data to a host are provided in the Slave module. The Slave module communicates with peripherals according to the command sent by the host, after verifying the address sent through the address byte.

In the Slave module implementation, the UART0 on-chip peripheral of the Z8 Encore! XP is initialized for multiprocessor mode by setting the UARTx Control 1 Register (UxCTL1) as described in Table 3.

Table 3. UARTx Control 1 Register (UxCTL1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MPM	MPE	MPBT	X	X	X	X	X
Reset	0	0	0	0	0	0	0	0

Note: X = Undefined.

Bit Position	Value	Description
7	MPM	**Multiprocessor Mode Select** Set this bit to 1 to enable Multiprocessor mode.
6	MPE	**Multiprocessor Enable** Set this bit to 1 to enable processor to process all address bytes.
5	MBPT	**Multiprocessor Bit Transmitter** Set this bit to 1. For the slave module implementation, this bit conveys no meaning.
4:0	—	Reserved.

The MPRx bit of the UARTx Status 1 Register (UxSTAT1) corresponds to the ninth bit of the received byte and indicates if the received byte is an address byte or a data byte. MPRx bit High (1) indicates that the received byte is an address byte with the address bit, and a MPRx bit that is Low (0) indicates that the received byte is a data byte. See Table 4.

Table 4. UARTx Status 1 Register (UxSTAT1)

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	MPRx	X	X	X	X	X	X	X
Reset	0	0	0	0	0	0	0	0

Note: X = Undefined.

Bit Position	Value	Description
7	MPRx	**Multiprocessor Receive** High indicates the status of the latest ninth bit received. Low indicates a reset.
6:0	—	Reserved.

The MPRx status bit is an indication to the receiver (Slave device) and reflects the actual status of the latest ninth bit (MPRx) received. When a Slave device reads from this register, the register resets this MPRx bit to 0.

In a Slave module, the software implementation for receiving address and data bytes and performing operations on the data (depending on the decoded command), involves initialization of the UART0 for communication in multiprocessor mode, setting the MPE bit in the U0CTL1 register to 1, and enabling the UART interrupts. Initializing ensures that the interrupts are generated by only those address bytes with address bits that match the Slave device address.

A UART interrupt occurs when an address byte is sent from the host. In the UART Interrupt Service Routine (ISR), the following steps are executed:

1) Read the status of the MPRX bit from the UART0 Status1 register (U0STAT1).

2) If the MPRX bit is 1, perform the following sub-steps:
   a) Read the byte received, which is the address byte.
   b) Decode the address, command, and length of data from the data in the U0RXD register.
   c) Check whether the decoded address matches the address of this Slave device.
   d) If the address matches this Slave address, set the MPE bit to 0 to receive UART interrupts when the host sends the data bytes to this Slave device.

3) If the MPRX bit is 0, perform the following sub-steps:
   a) Receive all the data bytes.
   b) Set the MPE bit to 1 to receive the next address from the host.
   c) Set the DATA_READY flag to 1.

In the main program:

1) Check the DATA_READY flag.

2) If the DATA_READY flag is set, perform the operation based on the decoded command information using the data sent by the host device. Set DATA_READY to 0.

For details of the implementation, see Appendix A—Flowcharts on page 9.

Testing the Application

The basic setup, equipment, and procedure for testing the Z8 Encore! XP 9-bit UART is presented in this section. The setup tests the implementation of the 9-bit mode to send a command from a host device to multiple Slave devices. The objective of this test command is to increase the duty cycle of a PWM signal from 25% to 50%.

Test Setup

The 9-bit UART is an on-chip peripheral for the Z8 Encore! XP MCU. Figure 6 displays the connection between the HyperTerminal application on a PC, Z8 Encore! XP Master device, and two Slave devices.

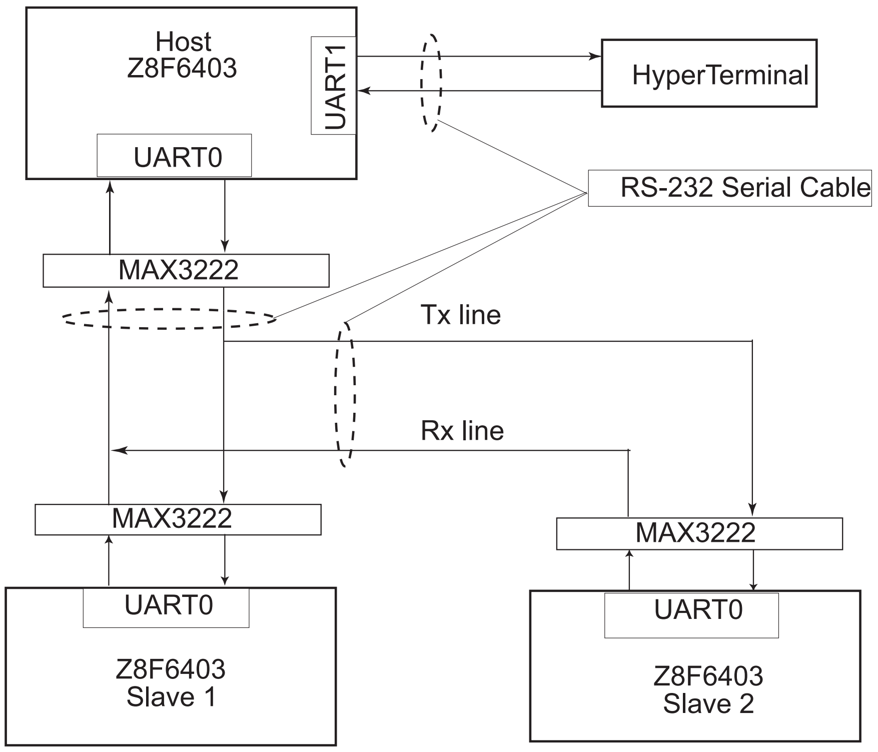

Figure 6. Test Setup for 9-bit UART Implementation

Equipment Used

The following equipment are used in the test setup:

- Z8ENCORE000ZCO Z8 Encore! XP Flash Microcontroller Development Kit (including a Z8 Encore! XP-based Evaluation Board)
- HyperTerminal application (PC-based)
- Oscilloscope

Procedure

Follow the steps below to test the 9-bit UART implementation:

1) Download the host code to the host Z8 Encore! XP device.

2) In the host Z8 Encore! XP device, UART1 is programmed to communicate with the HyperTerminal application, which is used for testing purposes.

3) Download the Slave code to the Slave Z8 Encore! XP devices.

4) Run the program in all of the Z8 Encore! XP devices.

5) In the Slave Z8 Encore! XP devices, Timer1 is programmed to output a PWM waveform of frequency 1 kHz with a duty cycle of 25%, which is available on the port pin PC1. An oscilloscope is connected to PC1 to observe the waveform.

6) Now, send the communication from the host Z8 Encore! XP over the network, and observe the results. For details the of transmission and results observed, see Table 5.

Table 5. Results of 9-bit Mode of Communication between One Host and Two Slave Devices

Test Case No	Transmitted Bytes—Ninth bit	Interpretation	Observation on PC1 of Slave 1	Observation on PC1 of Slave 2
1	0x25-1, 0x24-0, 0x00-0	On Slave 1, change PWM duty cycle from 25 to 50%.	Change in duty cycle from 25% to 50%.	No change.
2	0x45-1, 0x24-0, 0x00-0	On Slave 2, change PWM duty cycle from 25 to 50%.	No change.	Change in duty cycle from 25% to 50%.
3	0x05-1, 0x24-0, 0x00-0	On Slave 1and 2, change PWM duty cycle from 25 to 50%.	Change in duty cycle from 25% to 50%.	Change in duty cycle from 25% to 50%.

Summary

This Application Note describes a method of implementing a 9-bit mode of UART communication in a multidrop network with one Z8 Encore! XP MCU controlling up to eight Slave devices. The software implementation describes separate modules for these host and Slave devices.

The ninth bit of any transmitted byte is the MP bit. When set High, it indicates an address byte. When the ninth bit is set Low, it indicates a data byte. The address byte is divided into three distinct combinations of bits—three address bits, three command bits, and two bits to indicate the length of the data bits that follow the address byte.

A sample list of commands is included in this Application Note. The software listing implements the commands to set the PWM frequency and to change the duty cycle of the PWM signal

References

The document associated with Z8 Encore! and Z8 Encore! XP products available for download at www.zilog.com are listed below:

- Z8 Encore!® Flash Microcontroller Developer Kit User Manual (UM0146)
- Z8 Encore! XP® 64K Series Flash Microcontrollers Product Specification (PS0199)

Appendix A—Flowcharts

The flowchart for the Host module in the 9-bit UART implementation is displayed in Figure 7.

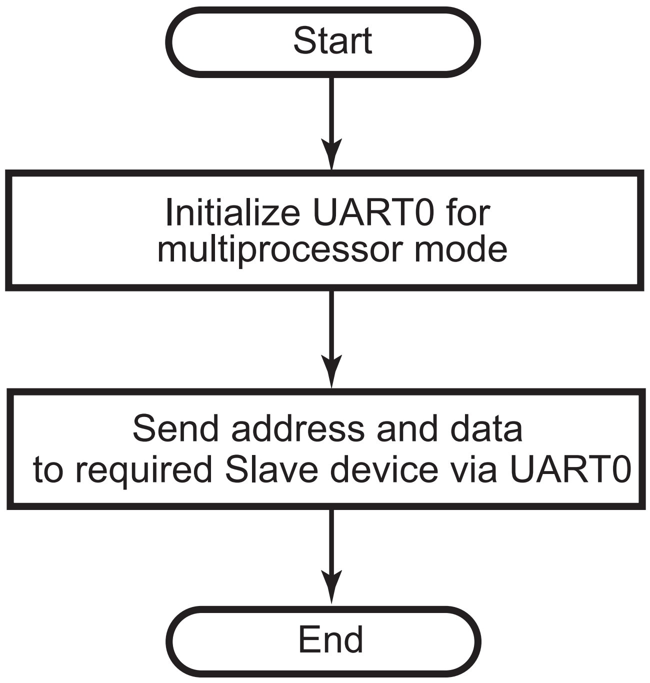
Start
Initialize UART0 for multiprocessor mode
Send address and data to required Slave device via UART0
End

Figure 7. Flowchart for Host Module

The flowchart for the Slave module in the 9-bit UART implementation is displayed in Figure 8

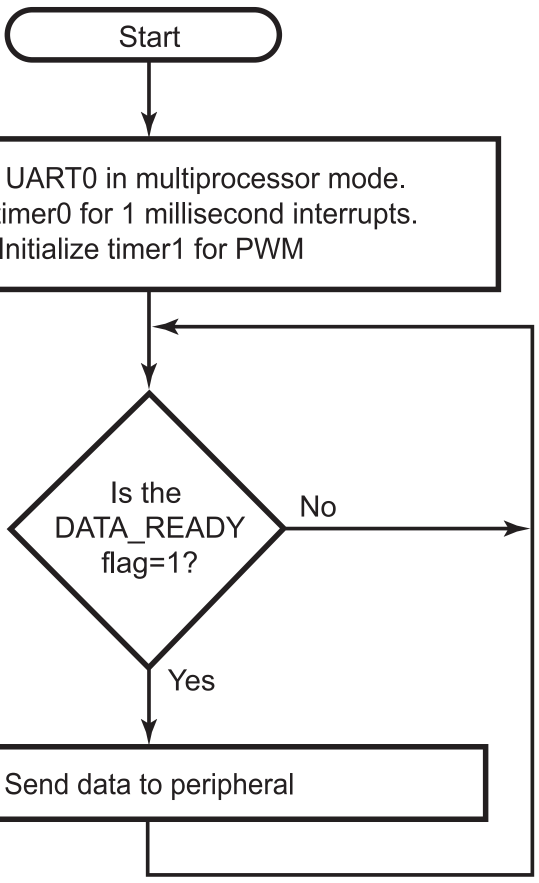
Start
Initialize UART0 in multiprocessor mode.
Initialize timer0 for 1 millisecond interrupts.
Initialize timer1 for PWM
Is the DATA_READY flag=1?
No
Yes
Send data to peripheral

Figure 8. Flowchart of Slave Module

Note: *The general-purpose timer APIs are used for initializing timer0 and timer1. For more details about timer APIs, refer to Using the Z8 Encore! Timer Application Note (AN0131).*

The flowchart for the UART0 Rx ISR is displayed in Figure 9.

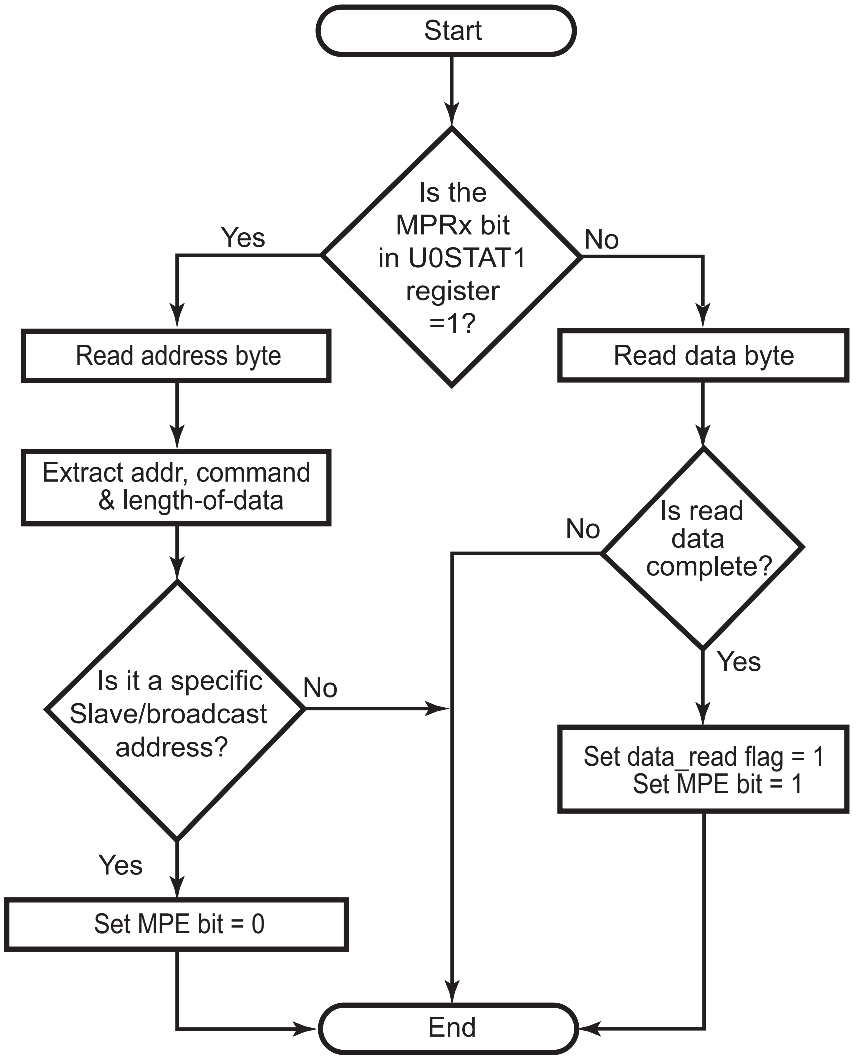

Figure 9. Flowchart for UART0 Rx ISR

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer