USING THE ZILOG Z8 CCP MCU AS AN I²C BUS MASTER

WHEN A STANDARDIZED PERIPHERAL BUS CONNECTION IS NEEDED, THE I²C SERIAL BUS CAN BE EASILY SIMULATED BY SOFTWARE CONTROL OF TWO Z8® CCP™ MCU I/O PINS.

INTRODUCTION

The Zilog Z8® Consumer Controller Processor (CCP™) family has many integrated features that simplify system design requirements in embedded applications. These on-chip peripheral features include Power-On Reset (POR), Low-Voltage Protection, Watch-Dog Timer (WDT), programmable I/O, comparators, multiple timer/counters (including external event counting and output waveform generation), and multiple external/internal interrupts. Some applications, however, may require additional external peripheral support, such as an I/O expander, specialty memory, LCD driver, or Analog-to-Digital (A-D) converter. To provide a standardized peripheral bus connection, a simple bidirectional, two-wire serial interface called the Inter-IC (I²C) bus may be used. While the Z8 CCP MCU family members do not possess dedicated on-chip I²C support hardware, this serial bus may be easily simulated by software control of two MCU I/O pins.

This application note provides an overview of the I²C bus interface and the 24CXXX EEPROM family, and details an application interfacing the Z8 CCP MCU (I²C Master) to a two-wire 24CXXX EEPROM (I²C Slave).

I²C BUS OVERVIEW

The I²C bus uses a two-wire interface consisting of a serial data line (SDA) and a serial clock line (SCL) to exchange information between devices connected to the bus. Each device on the bus has its own unique address and can operate as a transmitter or receiver (depending on its particular function). Devices are further characterized as Masters or Slaves. A “Master” is defined as a device that initiates, controls (generates all framing and clock signals), and terminates a transfer. A “Slave” is defined as any device addressed by a Master. Multiple Masters are allowed by I²C bus specifications because a bus arbitration and clock synchronization scheme is defined so that messages are not corrupted when more than one device attempts to take Master control. This procedure is also facilitated because all connections to the bus are wired-AND connections.

Both the SDA and SCL lines are bidirectional and are pulled up to the positive logic supply rail (via pull-up resistors, both lines High for the bus idle state). When in the output mode, the SDA and SCL lines are open-drain or open-collector. Data is exchanged 8 bits at a time and can be transferred at rates up to 100 Kbps (standard mode) or 400 Kbps (fast mode).

Data transfers on the I²C bus are controlled (and framed) via two unique bus states generated by the bus Master—START and STOP bit conditions. A START condition is defined as a High- to Low-level transition on SDA while the SCL line is High. A STOP condition is defined as a Low-to High-level transition on SDA while the SCL line is High (refer to Figure 1). Data must always be valid (stable) on the SDA line during the SCL High time and that it is only allowed to be changed during the SCL Low period (refer to Figure 2). Thus, one bit of data is transmitted for each SCL period.

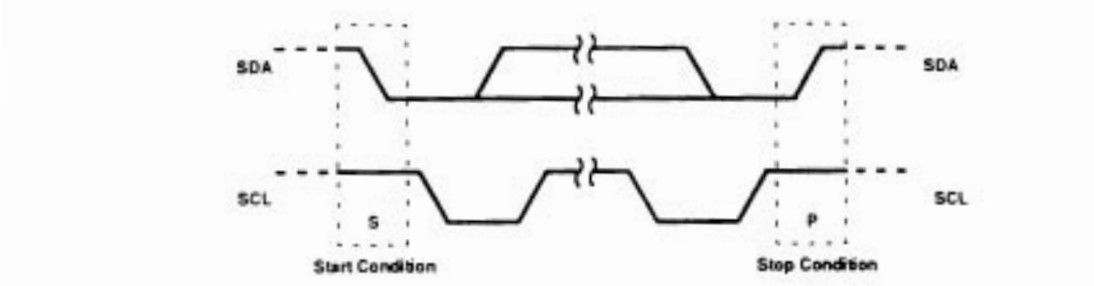

Figure 1. START and STOP Conditions

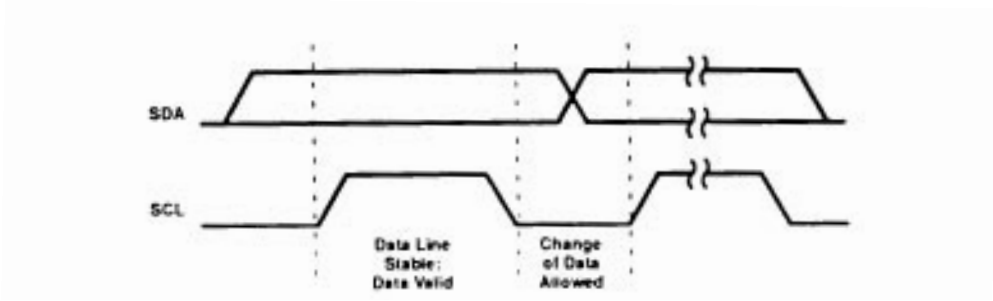

Figure 2. Valid Bit Data on the I²C Bus

Following the START condition, the first 8-bit byte sent in a bus message is a 7-bit Slave address field along with a data direction or R/W bit. (This discussion is limited to the 7-bit addressing mode.) The data direction bit (least significant bit) controls whether or not the Master will transmit (0 = write) or receive (1 = read) data from the addressed Slave. For every 8-bit byte exchanged, the most significant bit is always transmitted first. All 8-bit byte bus transactions (whether address, data, and so on) are followed by an acknowledge bit (refer to Figure 3). The acknowledge bit is a low-level signal placed on the SDA line by the receiving device (Master or Slave) during the Master transmitted acknowledge clock pulse (ninth High SCL clock pulse of the byte transmission).

If the receiver is unable to receive data (busy Slave receiver) or must signal the end-of-data condition (Master receiver), a non-acknowledge is sent (SDA High during the ninth High SCL clock pulse time, as shown in Figure 3). Following the START and Slave address transmission, data is exchanged between the Master and receiver as required. Upon exchange of the final byte and its acknowledge, the Master issues the STOP condition to end bus usage. (See Figure 4 for a sample I²C bus data transfer session.)

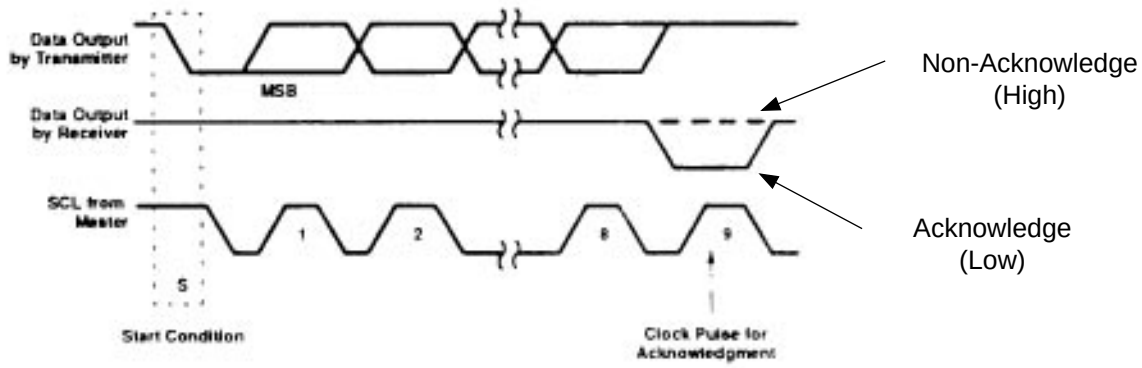

Figure 3. I²C Bus Acknowledge, Non-Acknowledge

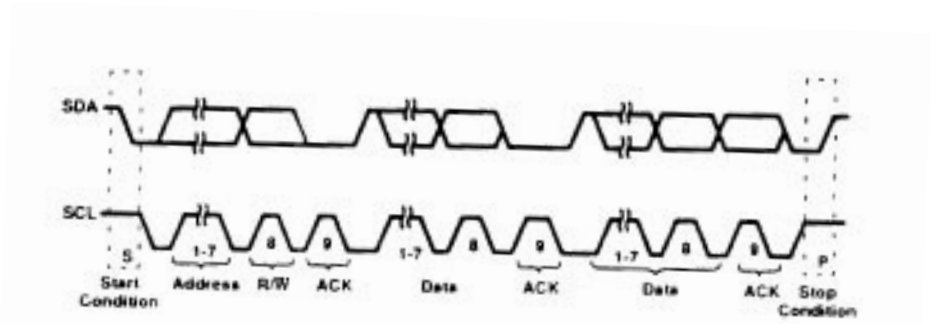

Figure 4. Sample I²C Bus Message

The previous general overview of I²C bus operation provides the foundation necessary to proceed with a serial EEPROM interface (a working knowledge of the I²C 7-bit addressing mode).

For complete details of all the various modes this bus supports, see the I²C Bus Specifications published by Philips Corporation, referenced at the conclusion of this app. note.

24CXXX SERIAL EEPROM OVERVIEW

The 24CXXX Family of two-wire serial EEPROM devices provide 256 to 2K (24C01A to 24C16) bytes of non-volatile storage and are I^2C bus compatible. The standard 8-pin PDIP package pinout is shown in Figure 5.

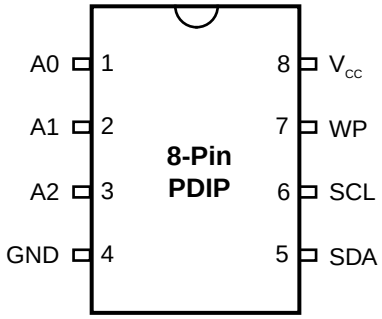

Figure 5. 24CXXX EEPROM 8-Pin PDIP Pinout

In addition to SDA and SCL, device control pins include:

- Write Protect (WP) Input. When this pin is grounded, normal read/write EEPROM accesses are permitted. When connected to V_{CC}, the EEPROM is write protected.
- A2, A1, A0: Device Address Inputs. All Ax address pins are hard-wired to fix the device's programmable software address.

EEPROM Addressing

Data is exchanged with the 24CXXX as a standard I^2C bus Slave device. After the START command, the first byte transmitted is the EEPROM's address byte. The upper nibble identifies the device as memory (per the I^2C specification). The lower nibble selects which 24CXXX is targeted for exchange (Ax, up to 8 devices may be on the bus) and/or which internal 256-byte block in the EEPROM (Px). (See Table 1 for address byte definition.)

Table 1. 24CXXX Device Name/Address Byte Definition

Device	No. of Bytes	Max. No. of Devs.	Page Write (No. of Bytes)	Device Address Byte (MSB to LSB)
24C01A	128	8	8	1010 A2 A1 A0 R/ W
24C02	256	8	8	1010 A2 A1 A0 R/ W
24C04	512	4	16	1010 A2 A1 P0 R/ W
24C08	1K	2	16	1010 A2 P1 P0 R/ W
24C16	2K	1	16	1010 P2 P1 P0 R/ W

R/W: 0 = Write; 1 = Read.
Ax = Hard-Wired Device Address.
Px = Internal Software Address. (Selects which 256-Byte Block to Address.)

After the device address is transmitted, the desired 8-bit EEPROM word address is sent. At this point the desired device operation—read or write—is carried out.

Write Operations

Two types of write operations (byte and page) are supported (see Figure 6). Byte operations allow a random EEPROM address to be written. Byte write operations require transmission of the following:

- START Condition (Master)
- EEPROM Device Address with R/W Bit = 0 (Master)
- Acknowledge Bit (EEPROM)
- Target EEPROM Word Address to be Written (Master)
- Acknowledge Bit (EEPROM)
- Data Byte to be Written (Master)
- Acknowledge Bit (EEPROM)
- STOP Condition (Master)

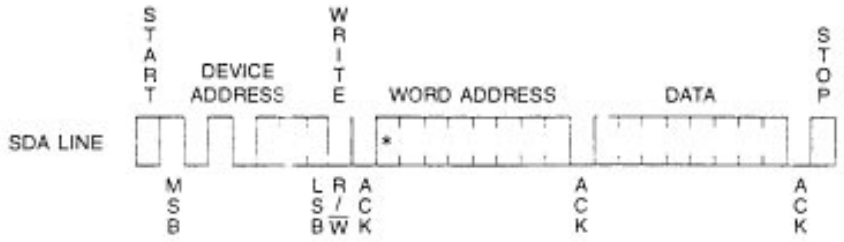

Figure 6. Byte Write Operation

Note that upon detecting the STOP bit, the EEPROM enters an internally timed write cycle to non-volatile memory. During this write cycle, all EEPROM inputs are disabled and the device will not respond to further bus activity until the write is complete. The Master device has two ways to determine when the EEPROM internal write cycle is completed: 1) timeout the maximum required write time (usually much longer than the device actually requires), 2) perform and acknowledge poll, which is the more time efficient way. This involves sending a start condition followed by the target EEPROM address. If the EEPROM acknowledges its address, the device write cycle is complete and the Master's desired operation may continue. (Otherwise, polling must be continued.)

Page write operations are identical to the byte write operations described previously, except that instead of writing one data byte, as many as 8 or 16 bytes (up to the maximum page write size the EEPROM present supports) may be continuously written after the EEPROM address and before the STOP bit transmission. (Refer to Table 1 and Figure 7.) During these page writes, the EEPROM automatically increments the address pointer between bytes.

Note: If the device's maximum page number is exceeded, the data will simply wrap around.

Once the STOP condition is received, those bytes updated will be written to the nonvolatile array. The great advantage of the page write feature is that only one write-cycle timing is consumed to write multiple bytes of data.

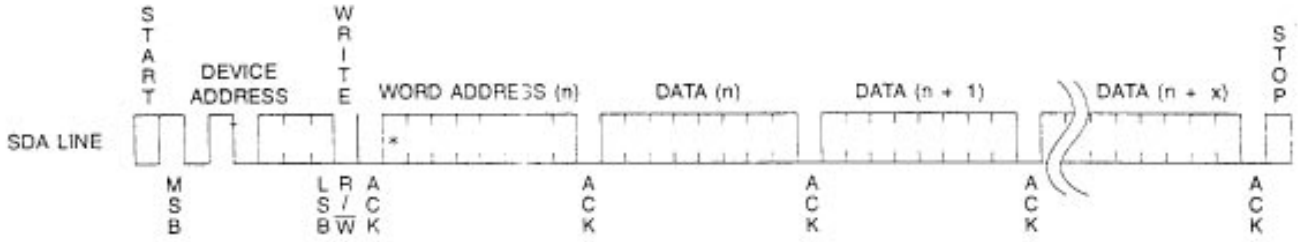

Figure 7. Page Write Operation

Read Operations

Three types of read operations are supported: Current Address, Random, and Sequential Read. Read operations begin just like write operations except the device address byte has the R/W bit set to 1.

For the Current Address Read mode, no EEPROM byte address is written as the data transmitted by the addressed Slave to the Master is read from the location of last access (incremented by one). This type read transmission sequence appears as follows:

- START Condition (Master)
- EEPROM Device Address with R/W Bit = 1 (Master)
- Acknowledge Bit (EEPROM)
- Data Byte to be Read (EEPROM: bytes sent are from the addressed Slaves last pointed to memory location incremented by 1)
- Non-Acknowledge Bit (Master)
- STOP Condition (Master)
- The Master signals a non-acknowledge and STOP condition to terminate data exchange with the Slave EEPROM (see Figure 8).

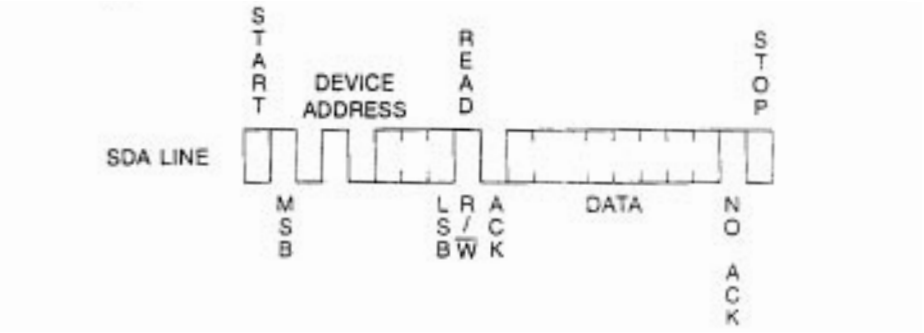

Figure 8. Current Address Read Operation

The Random Read mode is begun with a dummy byte write cycle (Master sends a START condition followed by the device address and target word address) followed by a Current Address Read mode cycle as described previously (see Figure 9).

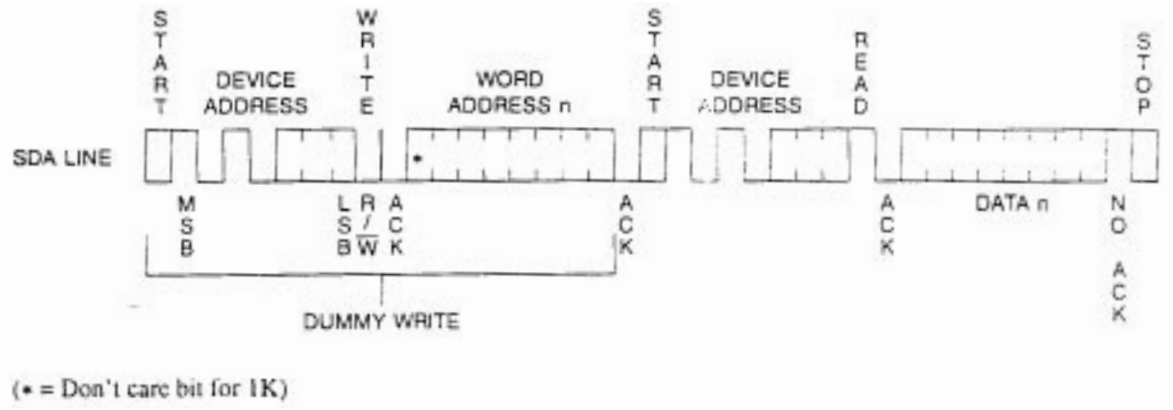

Figure 9. Random Read Operation

The Sequential Read mode is initiated with either a Current Address Read or Random Read. In this case, instead of the Master terminating the read after a single byte exchange (with a non-acknowledge), the Master responds with a valid acknowledge after each received data byte. This instructs the Slave EEPROM to continue the read operation and transmit out the next data byte. Sequential reads continue until terminated by the Master via issuance of a non-acknowledge on the last desired byte read followed by the STOP condition (see Figure 10).

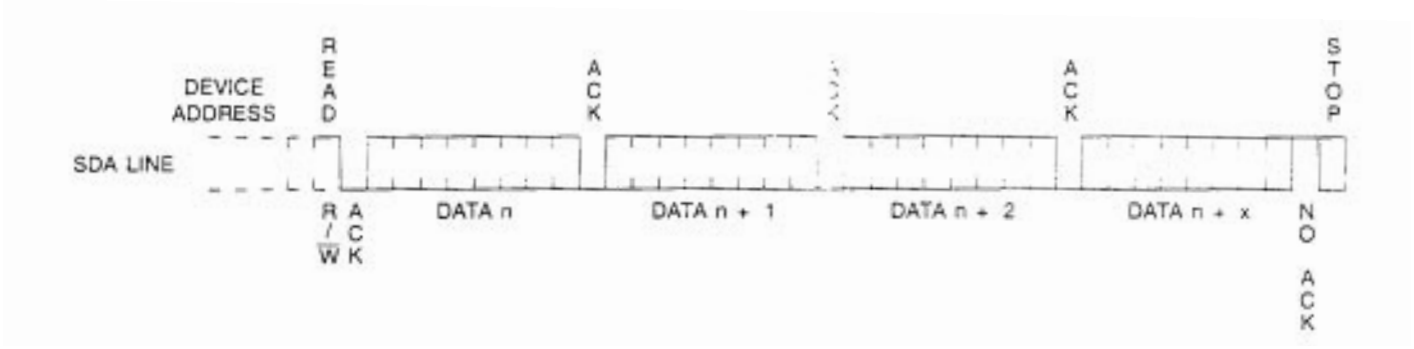

Figure 10. Sequential Read Operation

Z8 CCP MCU INTERFACE TO 24CXXX EEPROM

The hardware interface between the Z8® CCP™ MCU and a 24CXXX EEPROM is shown in Figure 11. The Write Protect (WP) input is tied to logic ground so that the MCU may write the EEPROM. Address line A0, A1, and A2 are also tied to logic ground.

Note: An external pull-up is required on the SDA line (open-drain output).

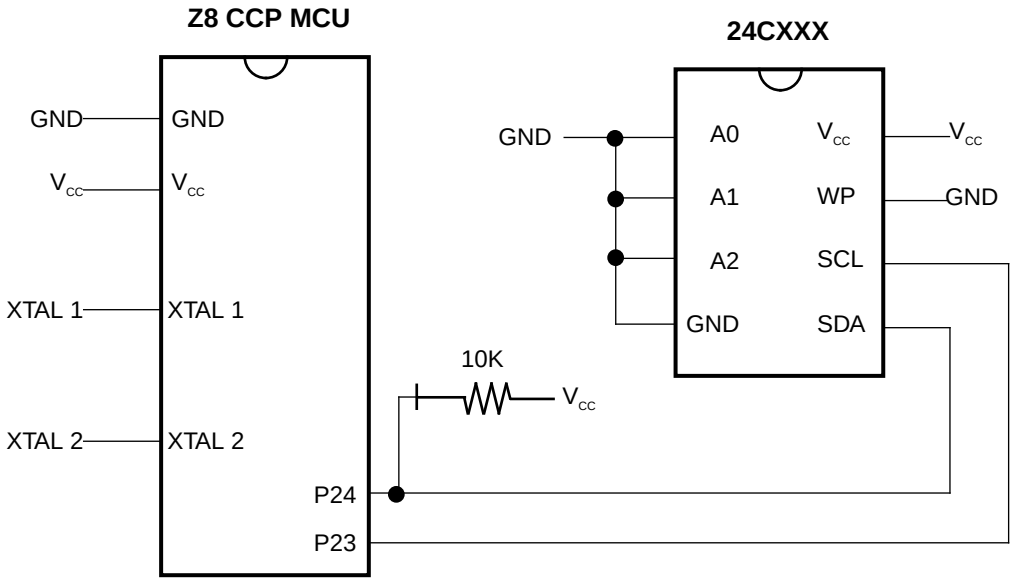

Figure 11. Generic Z8 CCP MCU-to-24CXXX Interface

ADDITIONAL REFERENCES

Appendix Summary

Refer to Appendix A for the 24CXXX Interface Software Flowchart and accompanying subroutine flowcharts.

Refer to Appendix B for applicable sample source code. The program listed in Appendix B will write/read the EEPROM in the stand-alone mode shown in Figure 11. The Z8 CCP MCU was clocked at 4 MHz for this sample code.

Note: Ensure that the selected EEPROM set-up and hold times are met at the application's Z8 clock frequency.

Secondary Sources

Atmel Nonvolatile Memory Data Book, Atmel Corporation, 1996.

I^2C Bus Specifications, Philips Semiconductors Microcontroller Data Handbook, Signetics Corp, 1992.

APPENDIX A: Z8 CCP MCU 24CXXX—INTERFACE SOFTWARE FLOWCHARTS

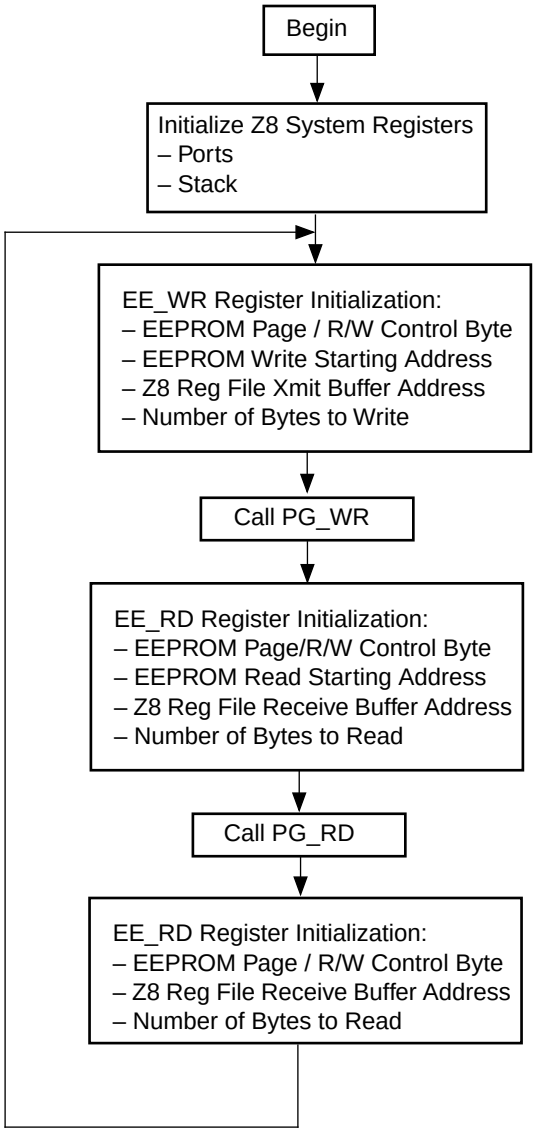

Figure 12. Z8 CCP MCU—24CXXX Interface Software Flowchart

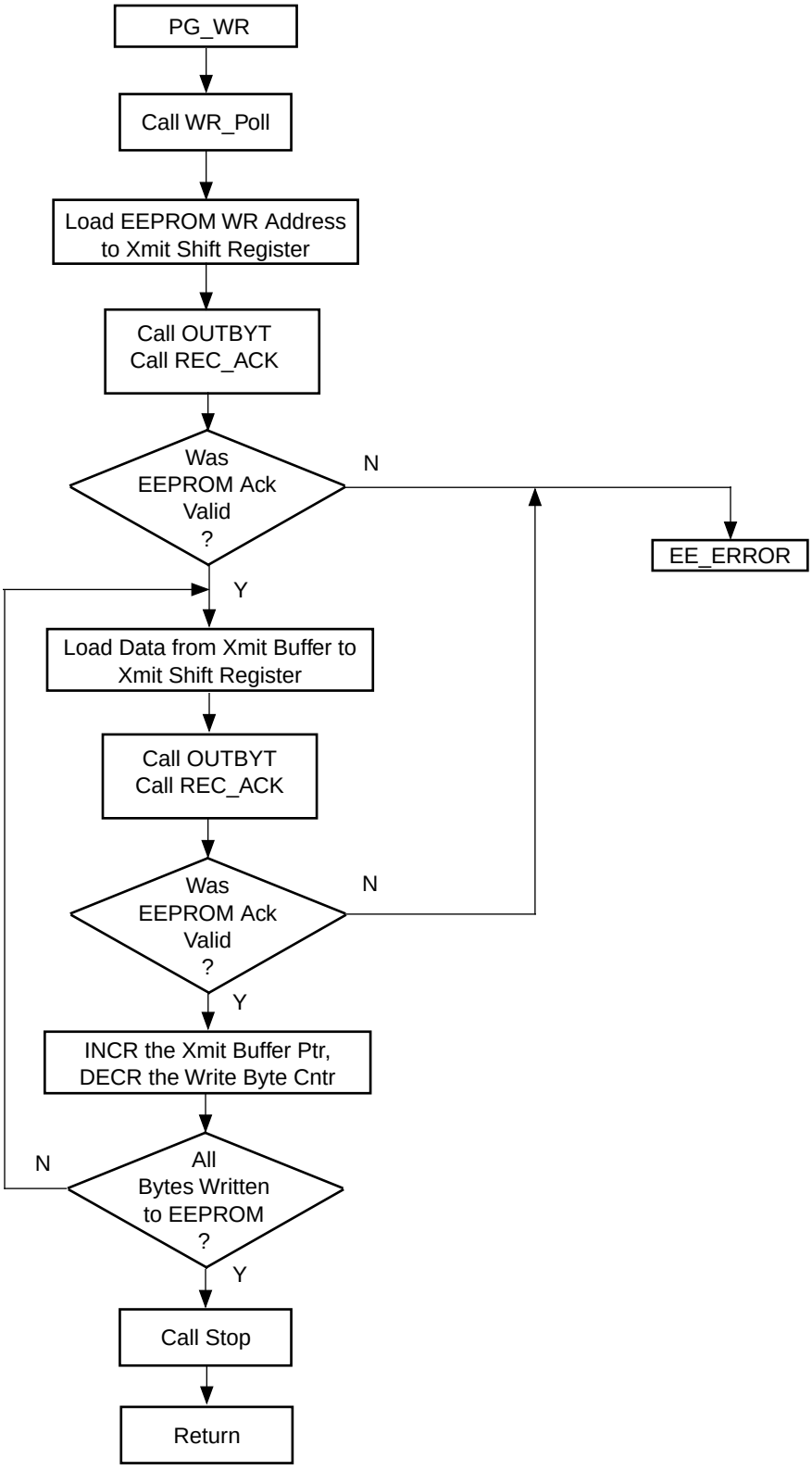

Figure 13. PG_WR Subroutine Flowchart

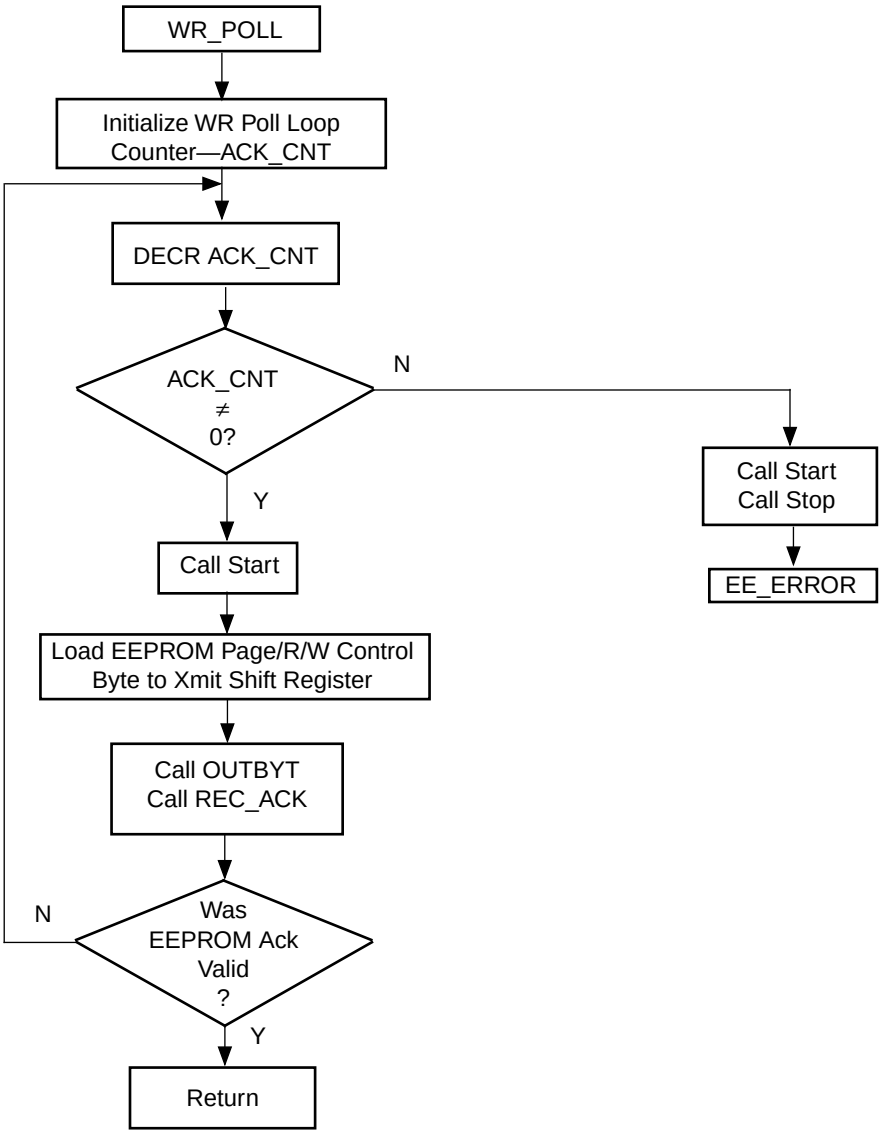

Figure 14. WR_Poll Subroutine Flowchart

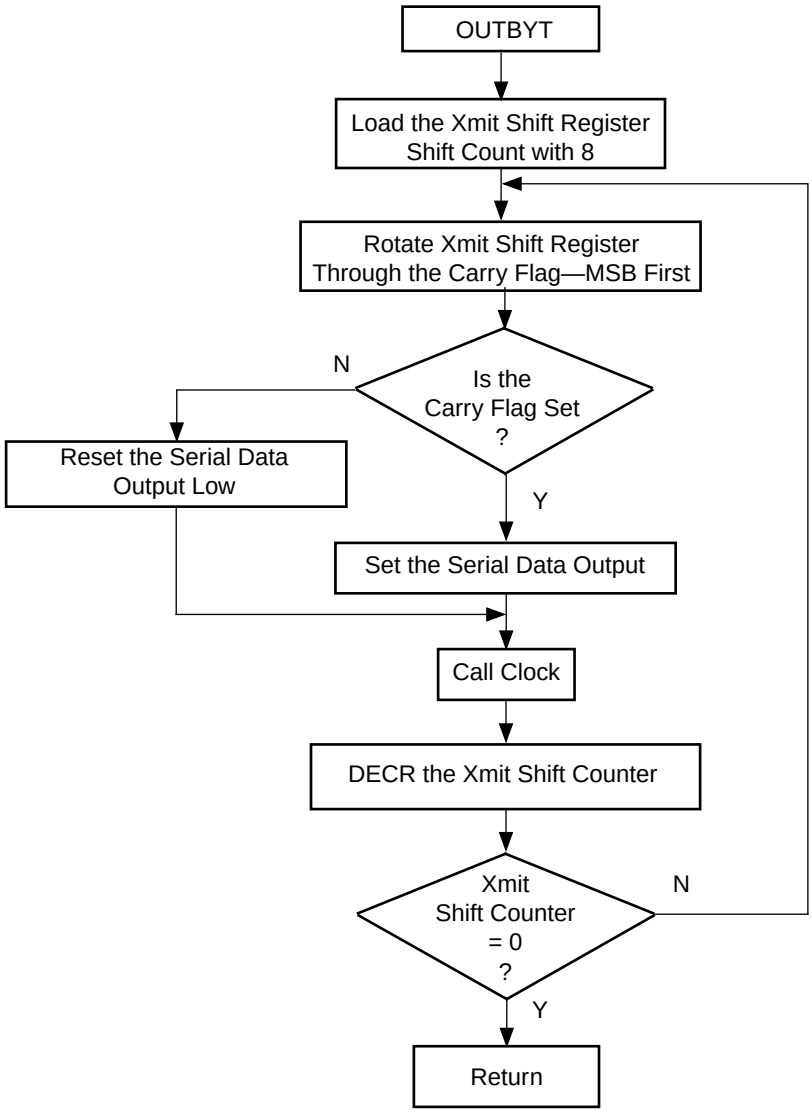

Figure 15. OUTBYT Subroutine Flowchart

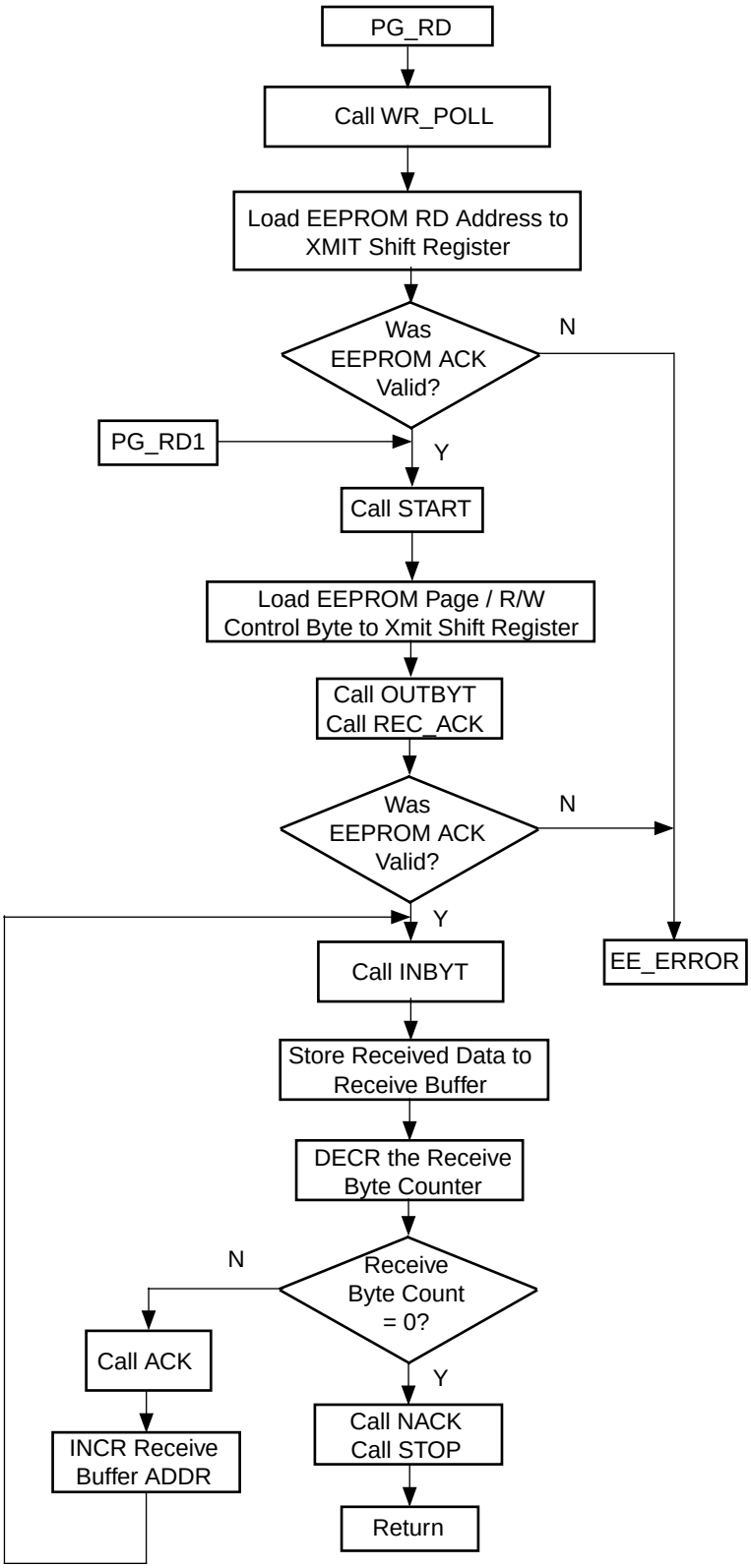

Figure 16. PG_RD Subroutine Flowchart

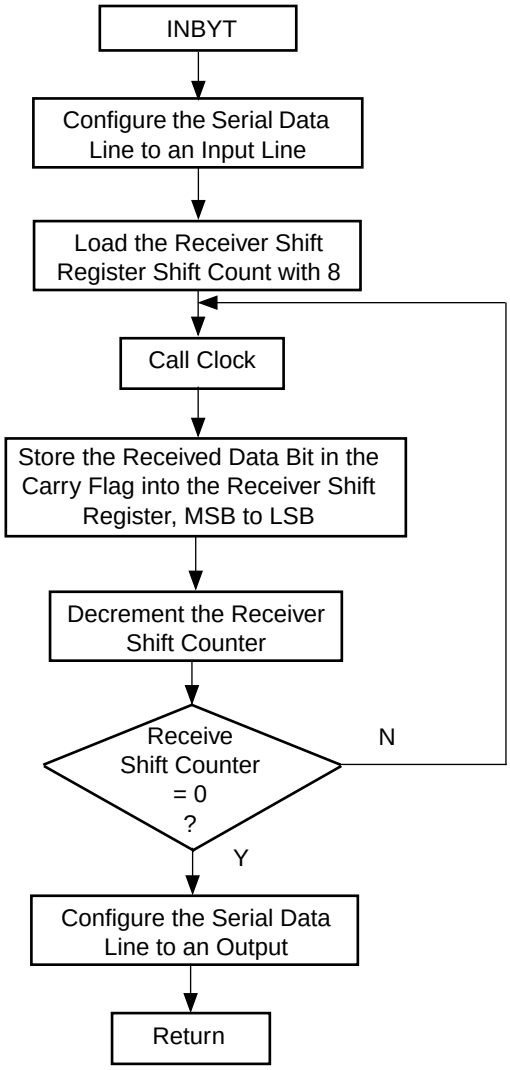

Figure 17. INBYT Subroutine Flowchart

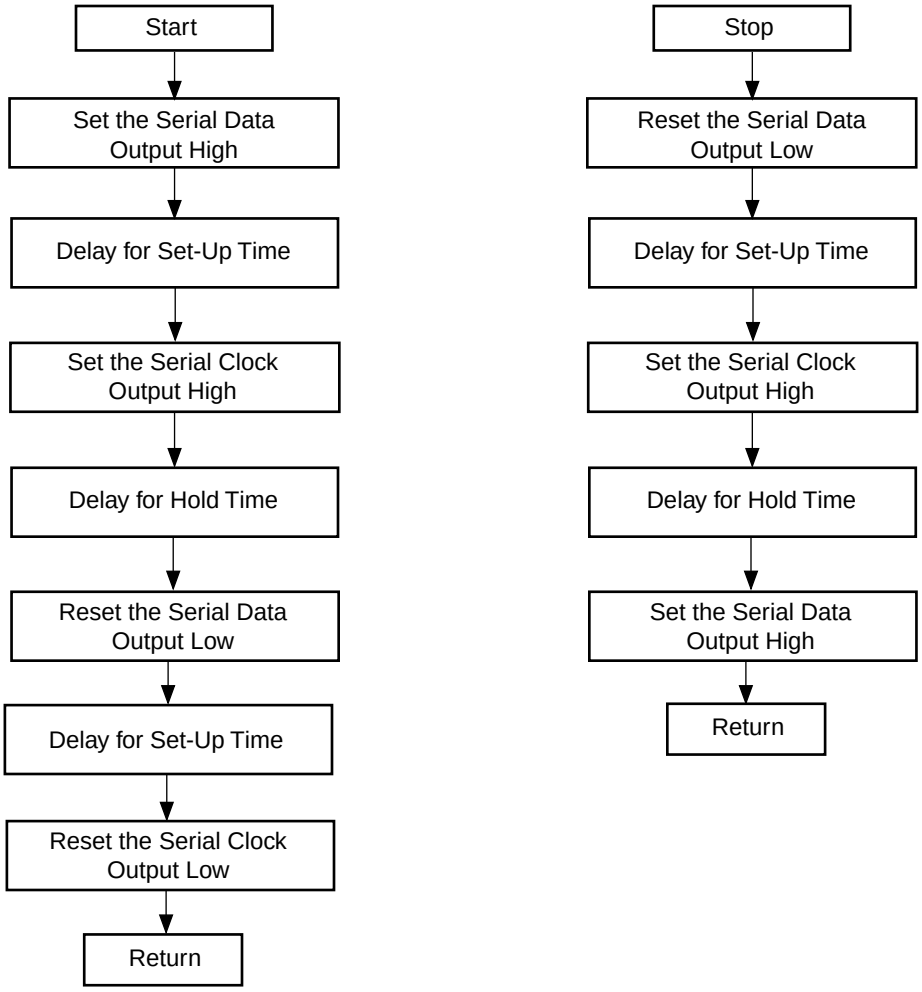

Figure 18. Start and Stop Subroutine Flowcharts

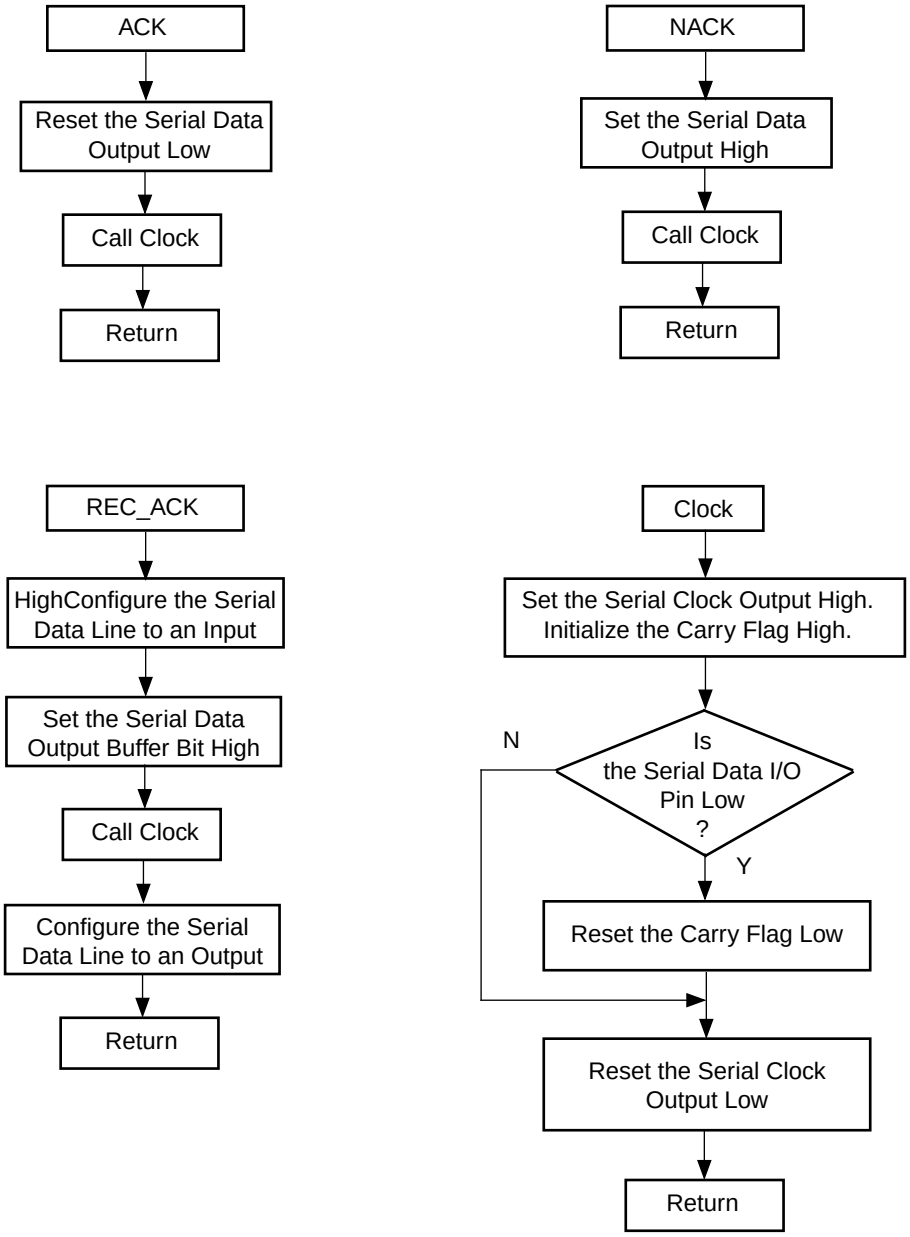

Figure 19. ACK, REC_ACK, NACK, and Clock Subroutine Flowcharts

APPENDIX B: Z8 MCU-TO-24CXXX INTERFACE SOFTWARE LISTING

(The Software Listing for the Internet copy of this app. note is located in separate downloadable file.)