AN14434

How to Implement MCX N236 USB to I3C Demo

Rev. 1.0 — 20 September 2024 Application note

Document information

Information	Content
Keywords	AN14434, MCX N236, USB, I3C, CDC Class
Abstract	This application note describes how to implement USB to I3C bridge function on MCX N236 microcontroller.

1 Introduction

Now in some applications such as PC and data center, the compute speed is increasing, which causes that the I^2C interface cannot fit with requirement. The I3C is an improved I^2C interface. It just needs two communication wires, but can work at speed up to 12.5 MHz, which is much higher than I^2C.

The USB interface is commonly used in many applications, which can speed up to 480 MHz in HS mode USB2.0 protocol. So a good interface can be used as a bridge to transfer the data from another interface such as USART, SPI, and I^2C to USB host. This application note provides a demo which uses the USB interface to bridge I3C interface. The MCX N236 microcontroller has one HS USB interface and two I3C interfaces.

1.1 MCX N236 I3C interface features

The I3C interface on MCX N236 supports:

- In-Band Interrupts (IBI): These interrupts move from target to controller without extra wires, and the controller knows which target sent the interrupt.
- Common Command Codes (CCC)
- Dynamic addressing
- Multi-controller and multi-drop
- Hot-Join (HJ)
- I^2C compatibility

The MCX N236 I3C IP supports all required and many optional features of the MIPI Alliance Specification for I3C, v1.0, and v1.1, except for ternary data rates (HDR-TSP and HDR-TSL).

2 MCX N236 USB to I3C demo introduction

In the MCX N236 USB to I3C demo, the USB device uses USB CDC virtual com class to communicate with PC host. You can use terminal tool to send a serial data to the control I3C interface. The terminal tool used in following contents is the `pzh-py-com` tool. Customers can download it from https://github.com/JayHeng/pzh-py-com.

This demo provides some commands such as dynamic address assign, direct write, direct read, write with register address, read with register address, IBI/Hot-join functions.

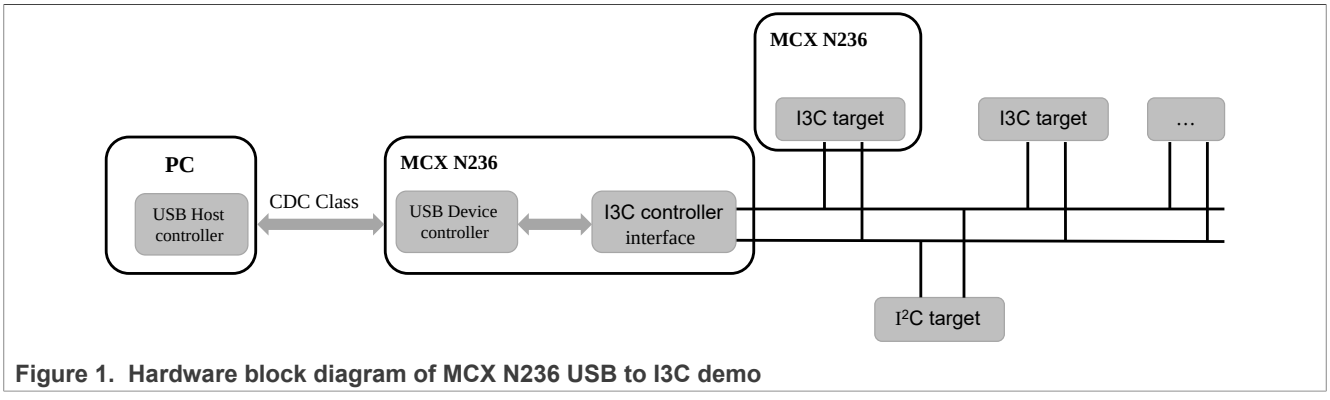

Figure 1. Hardware block diagram of MCX N236 USB to I3C demo

2.1 Hardware setup of MCX N236 USB to I3C demo

To perform the MCX N236 USB to the I3C demo, use two FRDM-MCXN236 boards, one as I3C controller and the other as I3C target. This demo uses `P1_16` (`I3C_SDA`) and `P1_17` (`I3C_SCL`) pins as the I3C function. Figure 2 shows the hardware connection. The connection wires must be as short as possible.

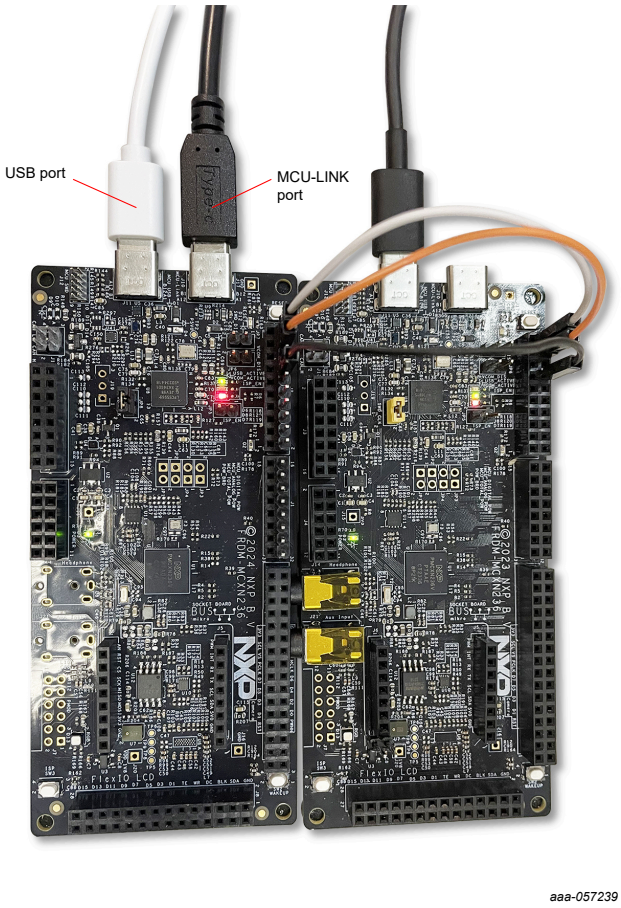

Figure 2. Hardware setup of MCX N236 USB to I3C demo

2.2 Software introduction of MCX N236 USB to I3C demo

The MCX N236 USB to I3C demo provides several commands for I3C configuration via the USB, which contains List DAA/direct write/direct read/read with register/CCC command send/Hot join and IBI commands. Customer can use these command to perform related I3C functions.

You can use a terminal tool which supports multi-strings send function to test this demo.

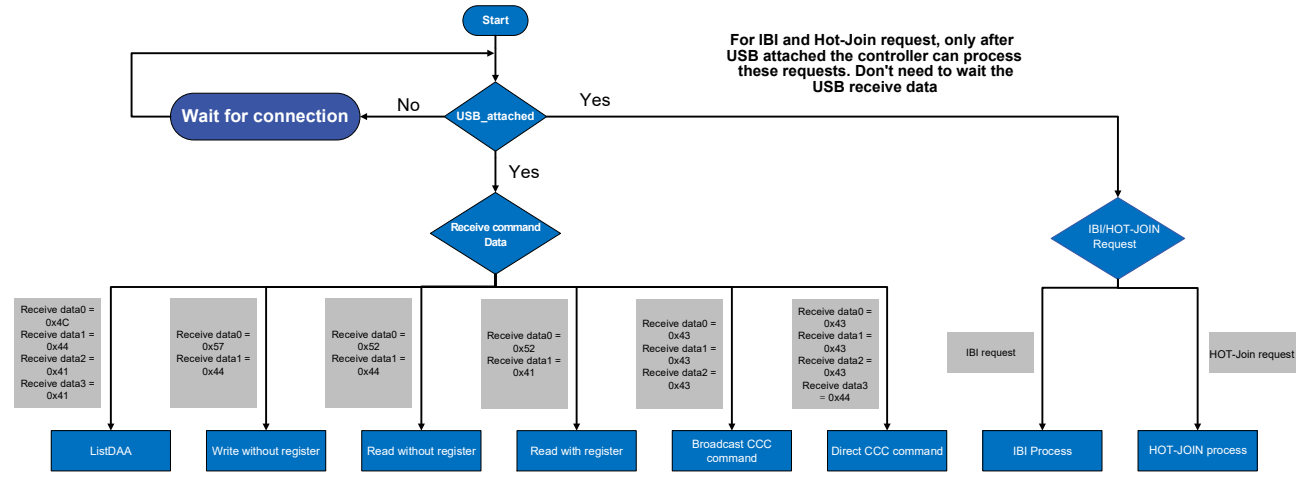

Figure 3. Software flow of MCX N236 USB to I3C demo

2.2.1 List DAA command

Figure 4 shows the I3C dynamic address assignment transaction. In the I3C protocol, the controller performs the dynamic address assignment to assign address for target devices which are connected. When using this command, find out how much devices are connected and give the dynamic addresses to be assigned.

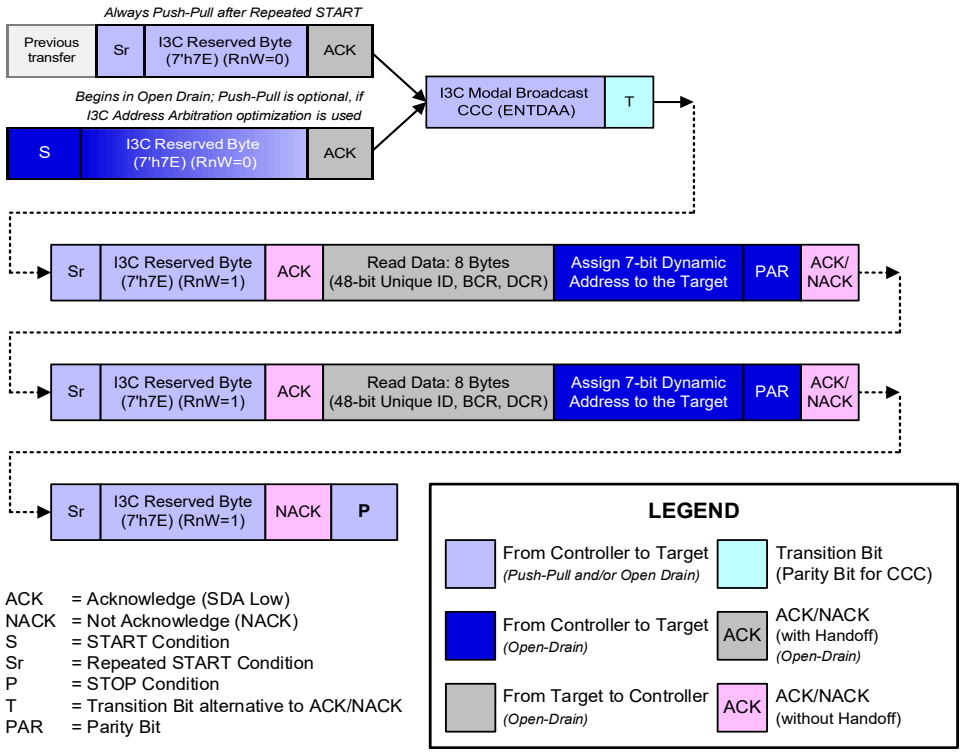

Figure 4. Dynamic address assignment transaction

Table 1 shows the structure of the List DAA command (send ASCII code).

Table 1. Structure of List DAA command

Supported commands								
ListDAA								
USB virtual com send	L	D	A	A	Address num	Dynamic address		
	0x4C	0x44	0x41	0x41				
USB virtual com receive	Vendor LSB	Vendor MSB	PartNumberInfo			BCR	DCR	
	—	—	—	—	—	—	—	—

In this demo, when the terminal sends the List DAA command, it receives the I3C target information feedback which contains the target vendor ID and BCR/DCR values, as shown in Figure 5.

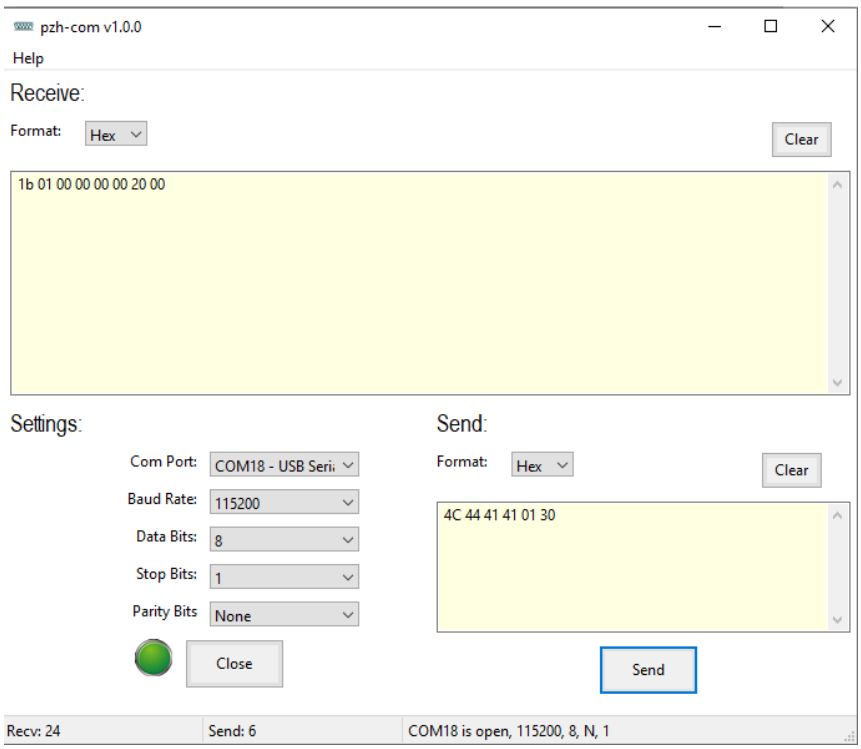

Figure 5. Terminal display of List DAA command

Figure 6 shows the I3C process timing of the List DAA command.

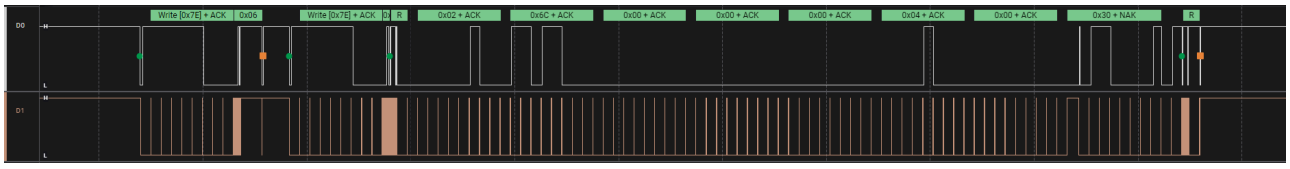

Figure 6. Process timing of List DAA command

2.2.2 Write without register address command

Table 2 shows the structure of the write without register address command (send ASCII code). When using this command, provide some information which contains target address/data size and write data.

Table 2. Structure of write without register address command

Supported commands								
Write without register address								
USB virtual com send	W	D	Slave address	Data size	Data (N byte)			
	0x57	0x44						
USB virtual com receive	O				K			
	0x4F				0x4B			

Figure 7 shows the OUT and IN structures of the write without register address command on the terminal. After the command finishes, the terminal receives OK (0x4F, 0x4B) characters.

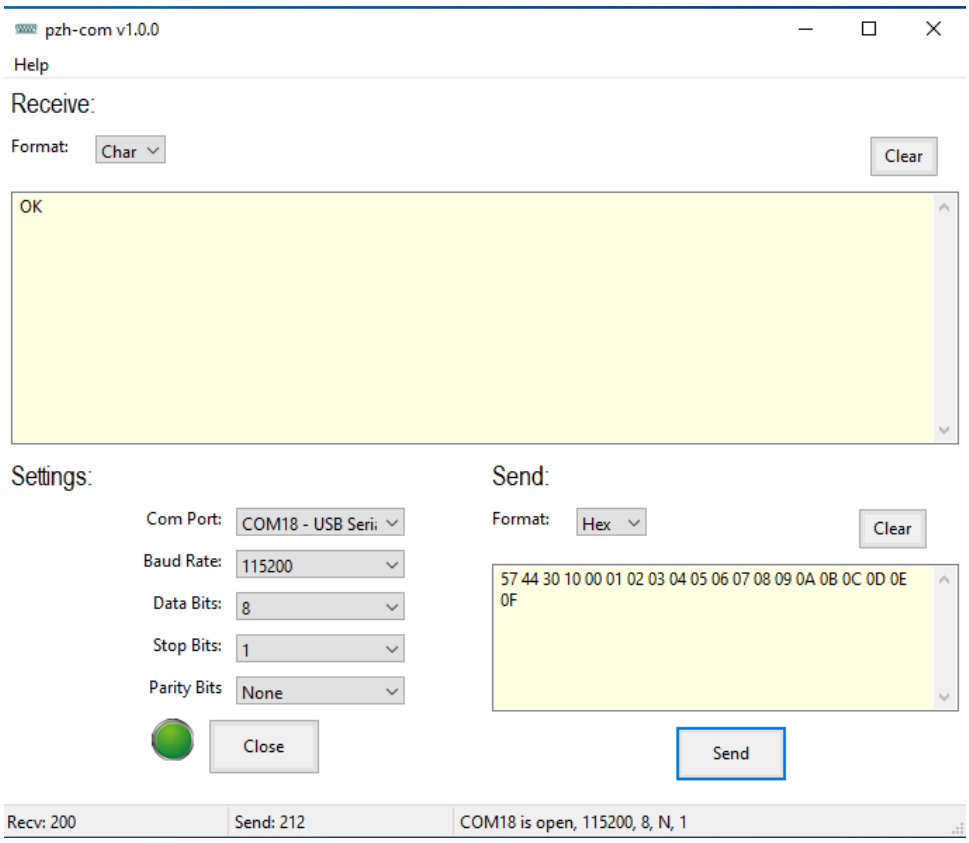

Figure 7. Terminal display of the write without register address command

Figure 8 shows the I3C process timing of the write without register address command

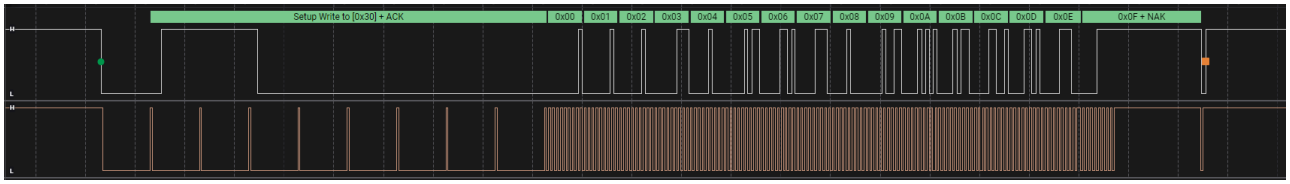

Figure 8. Process timing of write without register address command

2.2.3 Read without register address command

Table 3 shows the structure of the read without register address command (send ASCII code). When using this command, provide some information which contains target address and read data size.

Table 3. Structure of read without register address command

Supported commands								
Read without register address								
USB virtual com send	R	D	Slave address	Data size				
	0x52	0x44						
USB virtual com receive	ReadBack data (N bytes)							
	—	—	—	—	—	—	—	—

Figure 9 shows the OUT and IN structures of the read without register address command on the terminal. After the command finishes, the terminal receives the data sent by the target.

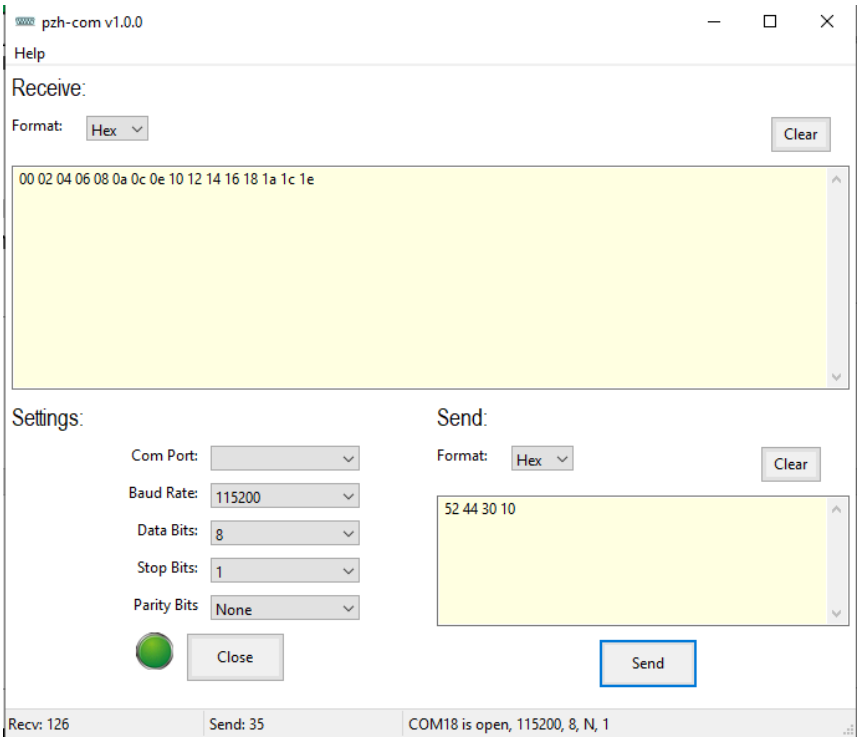

Figure 9. Terminal display of read without register command

Figure 10 shows the I3C process timing of the read without register address command.

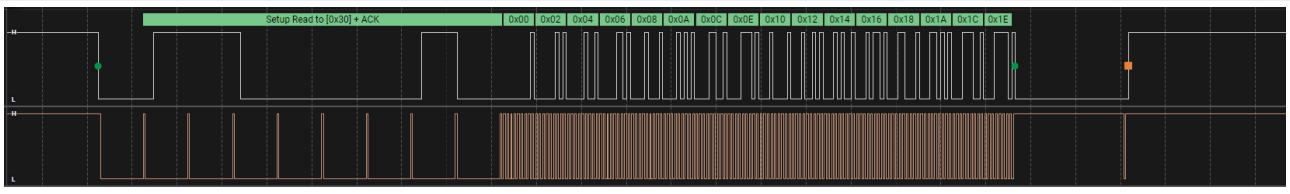

Figure 10. Process timing of read without register address command

2.2.4 Read with register address command

Table 4 shows the structure of read with register address command (send ASCII code). When using this command, provide some information which contains target address/register address and read data size.

Table 4. Structure of read without register address command structure

Supported commands								
Read with register address								
USB virtual com send	R	A	Slave address	Register address	Data size			
	0x52	0x41						
USB virtual com receive	ReadBack data (N bytes)							
	—	—	—	—	—	—	—	—

Figure 11 shows the OUT and IN structures of the read with register address command on the terminal. After command finished, the terminal will receive register data which were sent by target.

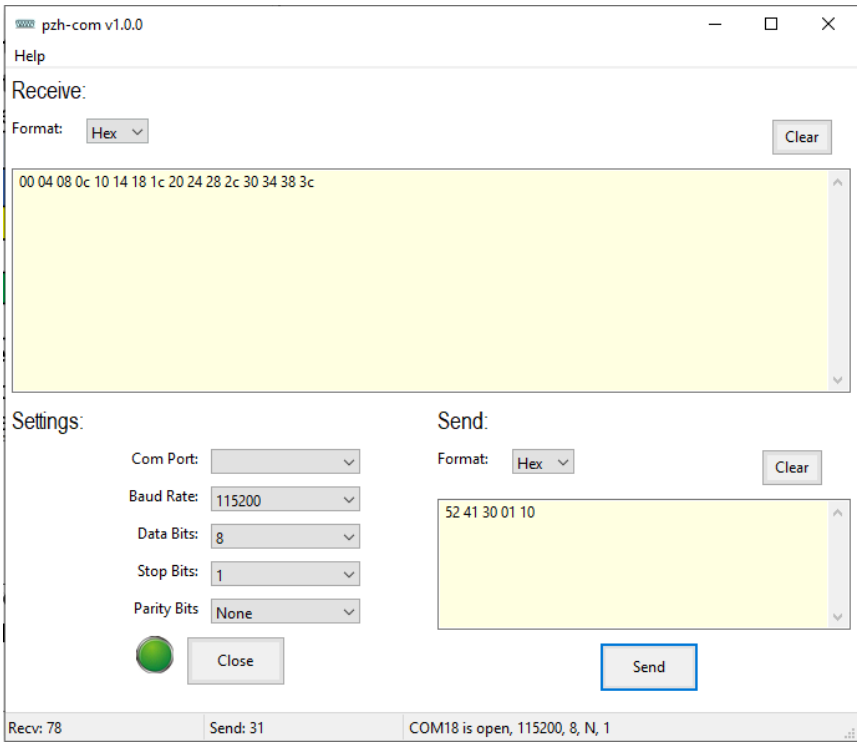

Figure 11. Terminal display of read with register address command

Figure 12 shows the I3C process timing of the read with register address command.

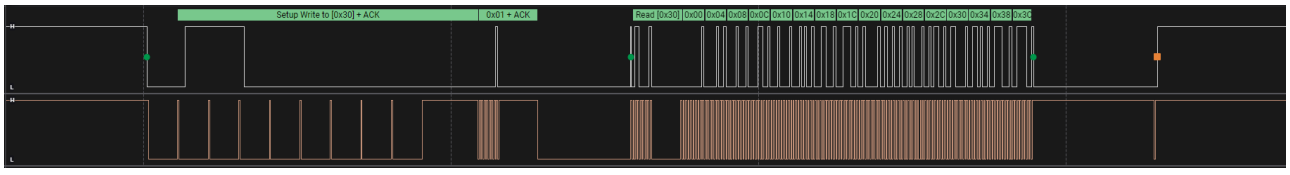

Figure 12. Process timing of read with register address command

2.2.5 CCC broadcast command

Figure 13 shows the CCC broadcast frame format. When sending the CCC broadcast command, the slave address is 0x7E. The I3C specification provides several CCC broadcast commands such as ENTDAA used to make I3C target in the DAA state. The ENTHDR0 command can be used to inform the targets act as HDR-DDR mode.

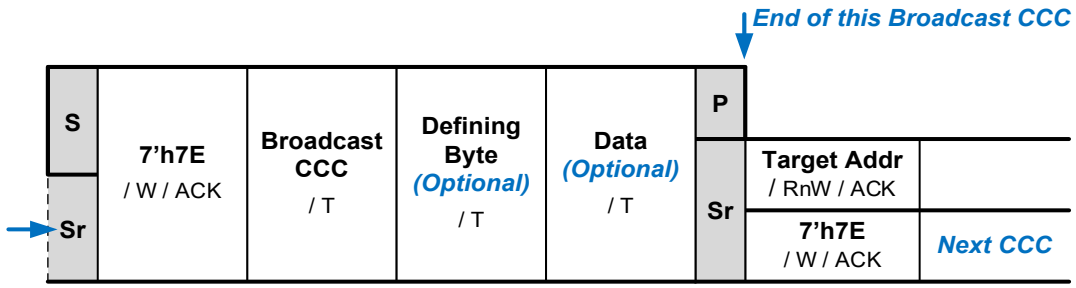

Figure 13. General frame format of CCC broadcast

Table 5 shows the structure of the CCC broadcast command (send ASCII code).

Table 5. Structure of CCC broadcast command

Supported commands								
CCC broadcast								
USB virtual com send	C	C	C	Command code				
	0x43	0x43	0x43					
USB virtual com receive	O				K			
	—	—	—	—	—	—	—	—

Figure 14 shows the OUT and IN structures of the CCC broadcast command on the terminal. After the CCC command is sent, the terminal receives the OK (0x4F, 0x4B) character.

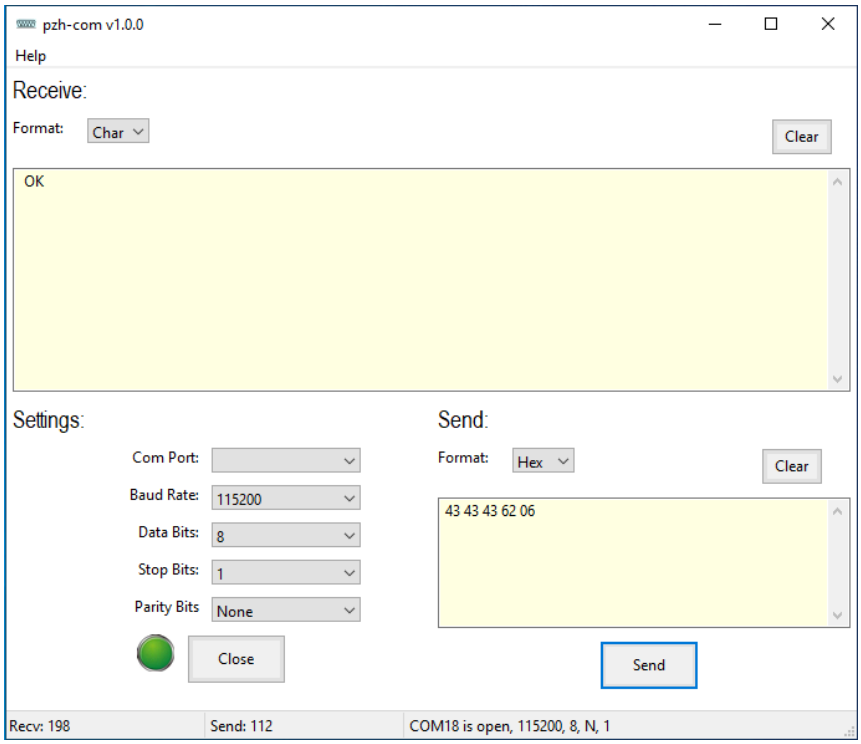

Figure 14. Terminal display of CCC broadcast command

Figure 15 shows the I3C process timing of the read with register address command.

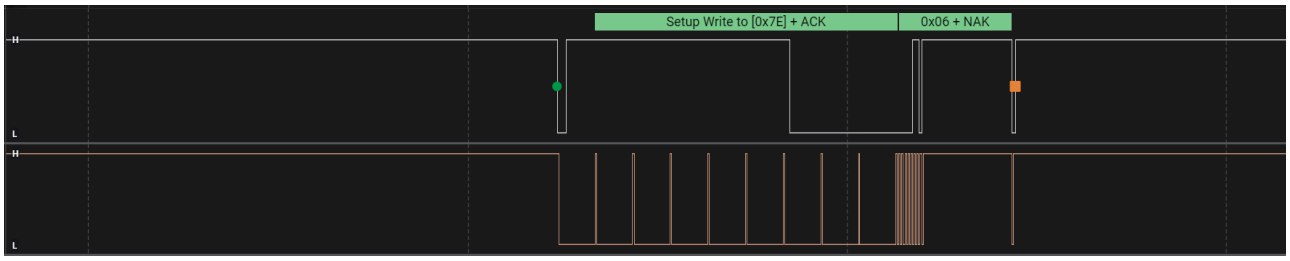

Figure 15. Process timing of CCC broadcast command

2.2.6 CCC direct send command

Figure 16 shows the general frame format of CCC direct send command. The I3C specification provides several CCC direct commands such as `GET_BCR` used to get the target BCR value. The `GET_STATUS` command can be used to get the target status value.

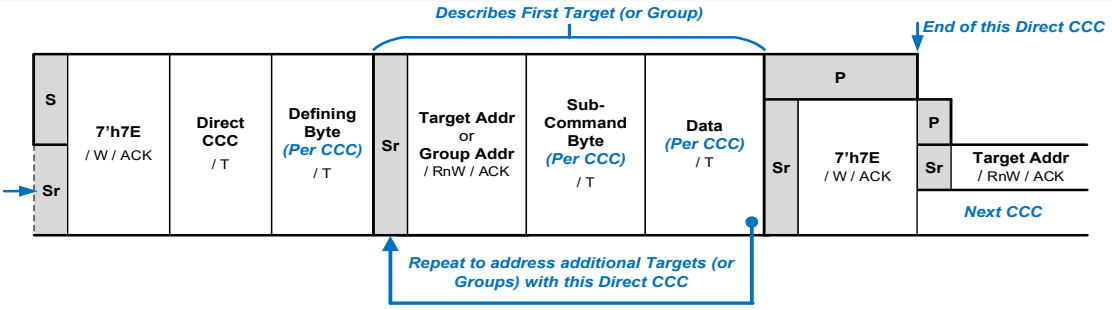

Figure 16. General frame format CCC Direct send command

Table 6 shows the structure of the CCC direct send command (send ASCII code). When using this command, provide the CCC direct command provided by the I3C specification such as `GET_BCR` (`0x8E`) and the target address.

Table 6. Structure of CCC broadcast command

Supported commands								
CCC direct								
USB virtual com send	C	C	C	D	Command code			
	`0x43`	`0x43`	`0x43`	`0x44`				
USB virtual com receive	Command send back data (such as BCR/DCR, DEVICE STATUS)							
	—	—	—	—	—	—	—	—

Figure 17 shows the OUT and IN structures of the CCC direct send command on the terminal. After the command finishes, the terminal receives corresponding data related to the send command such as the target BCR value.

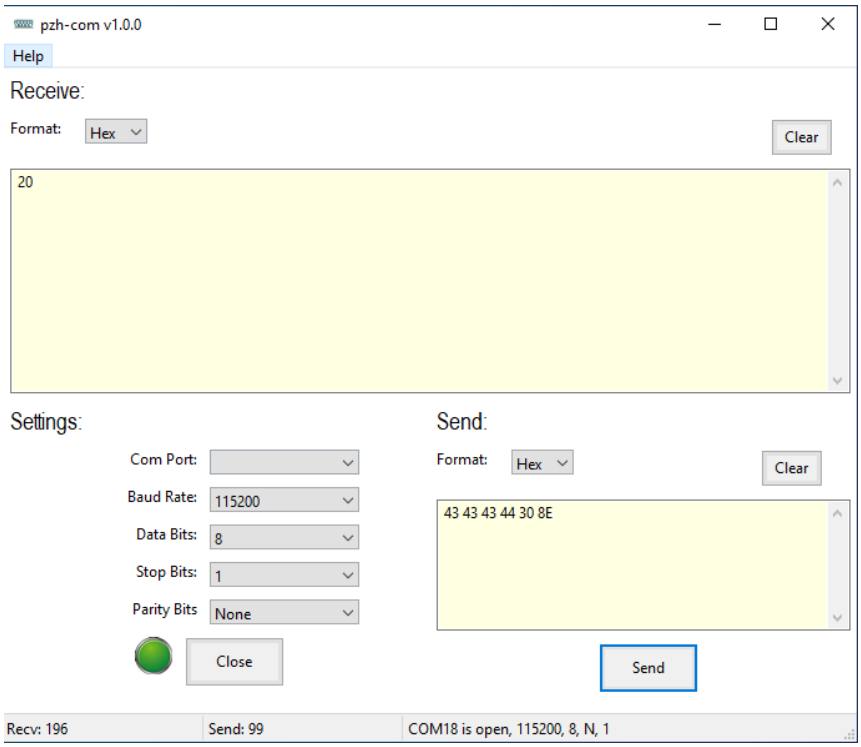

Figure 17. Terminal display of CCC direct send command

Figure 18 shows the I3C process timing of the CCC direct send command.

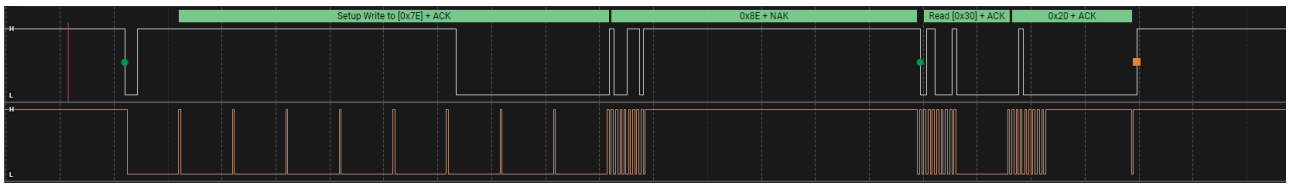

Figure 18. Process timing of CCC direct send command

2.2.7 IBI command

Figure 19 shows the IBI (Target Interrupt) request FSM. When the target generates the IBI signal, the controller performs theIBI response and the target sends the IBI data.

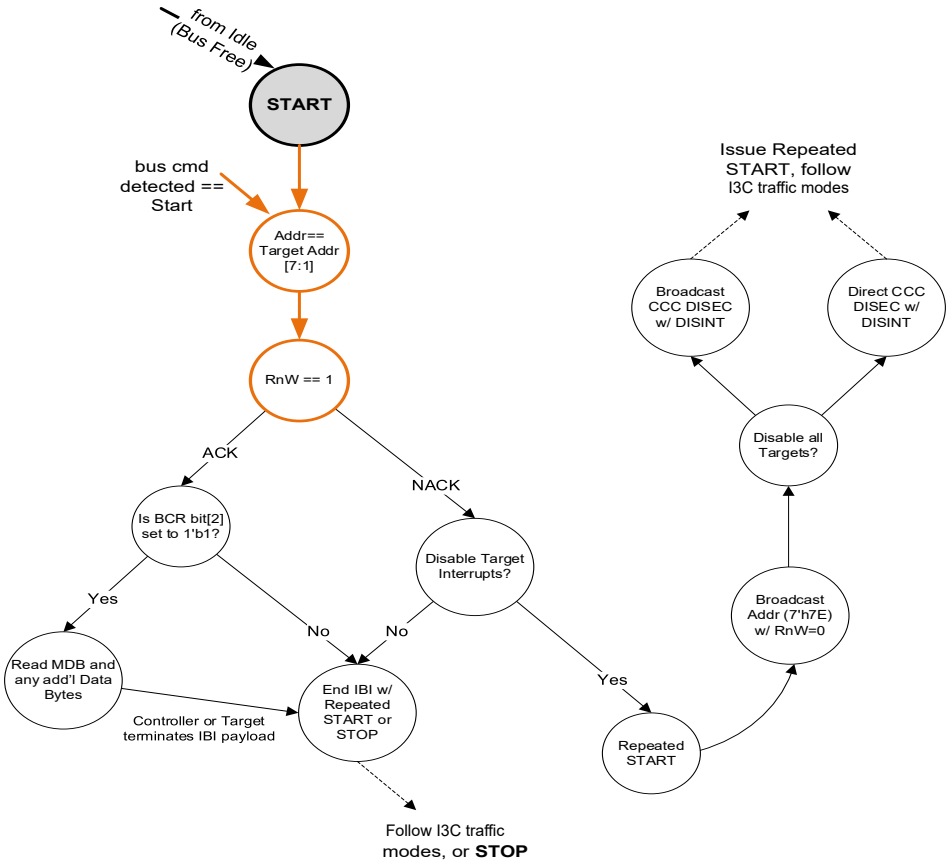

Figure 19. IBI (Target Interrupt) request FSM

Table 7 shows the structure of the IBI command. Actually, no data is sent for IBI response. When USB device connects to host, the I3C target can send IBI signal at any time. The USB host receives the IBI data when there is an IBI signal generated.

Table 7. Structure of IBI command

Supported commands								
IBI								
USB virtual com send	DO NOT SEND DATA (when there is IBI generated, virtual com receives IBI data)							
	—	—	—	—	—	—	—	—
USB virtual com receive	IBI DATA send by slave							
	—	—	—	—	—	—	—	—

Figure 20 shows the IN structure of the IBI command on the terminal. When the target generates an IBI signal, the terminal receives the IBI data sent by the target.

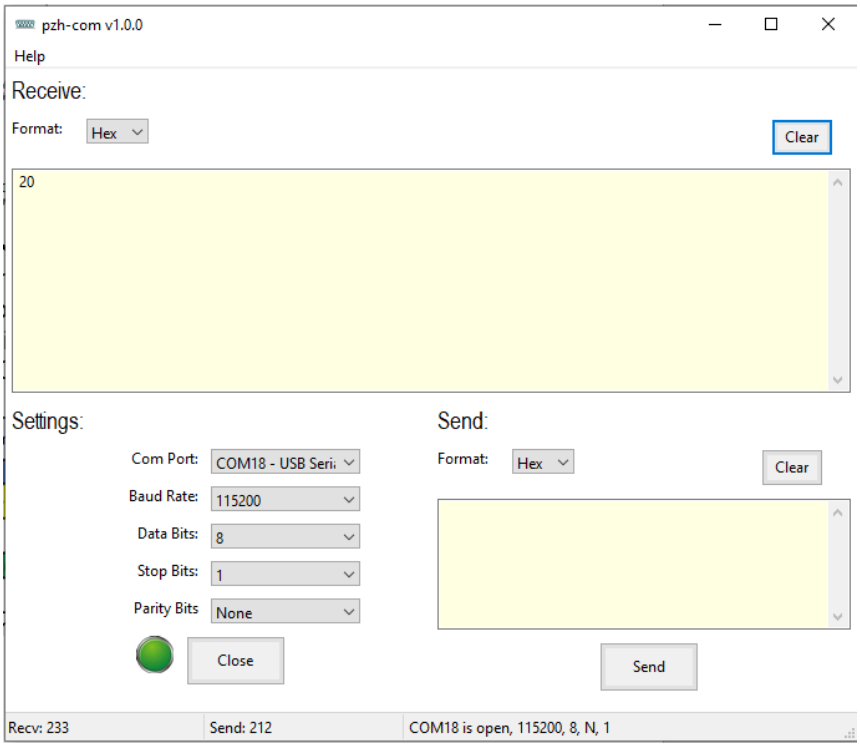

Figure 20. Terminal display of IBI command

Figure 21 shows the I3C process timing of the IBI command.

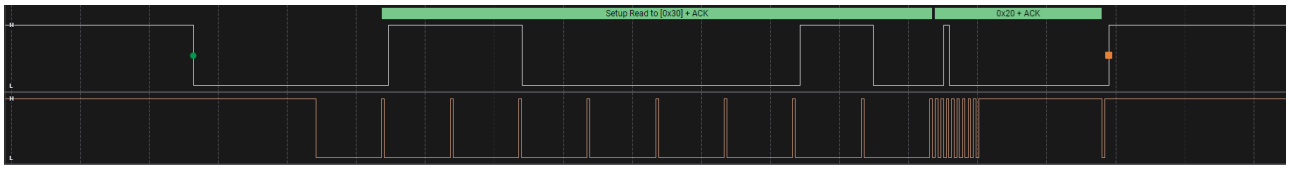

Figure 21. Process timing of IBI command

2.2.8 Hot-Join command

Figure 22 shows the Hot-Join FSM process by controller. After the target sends Hot-Join (`0x02`) signal, the controller processes the dynamic address assignment after it ACKed the Hot-Join request.

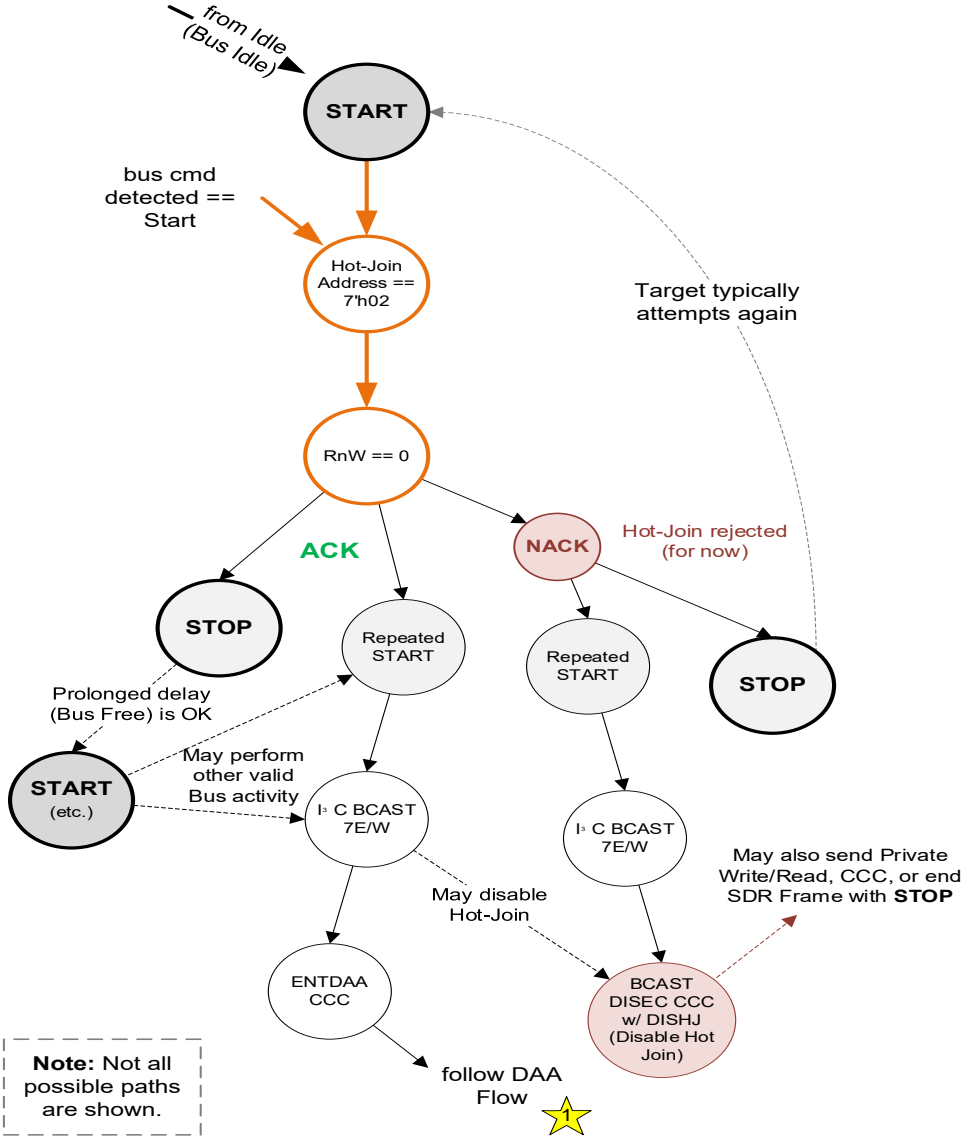

Figure 22. Hot-Join FSM

Table 8 shows the structure of the Hot-Join command, same with the IBI command, the Hot-Join signal was generated by target. When USB device connected to host, I3C target can send Hot-Join signal at any time. And the controller assigns the dynamic address after detecting the Hot-Join signal. The USB host receives the target vendor ID/Part Number Info/BCR/DCR and slave address same with the List DAA command feedback.

Table 8. Structure of Hot-Join command

Supported commands								
HotJoin								
USB virtual com send	DO NOT SEND DATA (when there is Hot-Join generated, the virtual com receives the IBI data)							
	—	—	—	—	—	—	—	—
USB virtual com receive	Vendor LSB	Vendor MSB	PartNumberInfo			BCR	DCR	slaveAddress
	—	—	—	—	—	—	—	—

Figure 23 shows the IN structure of the Hot-Join command on the terminal. When the target generates an Hot-Join signal, the terminal receive the targets information data which contains vendor ID/Part Number Info/BCR/DCR and slave address.

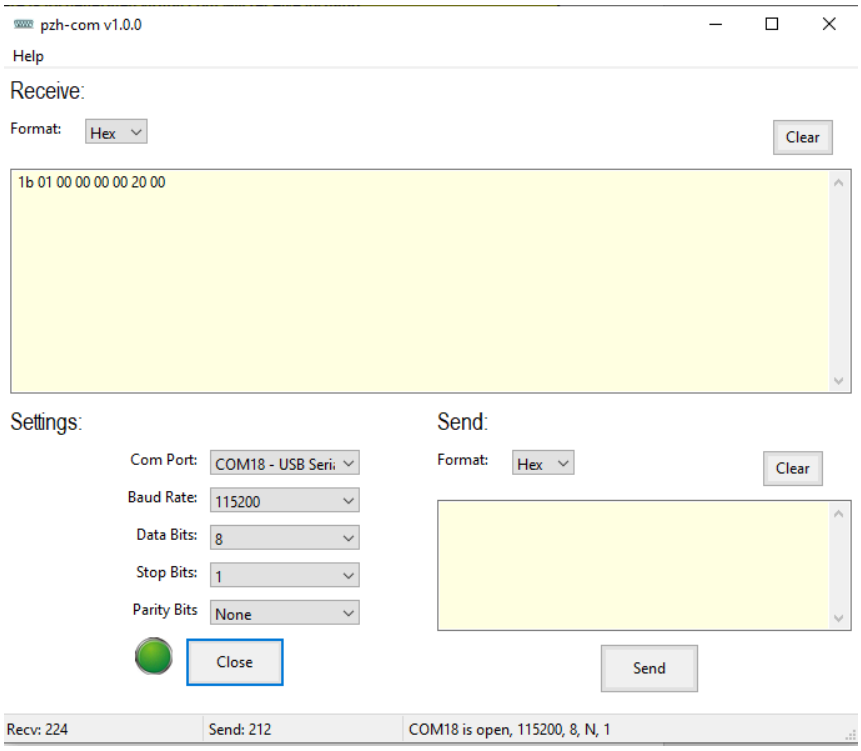

Figure 23. Terminal display of Hot-Join command

Figure 24 shows the I3C process timing of the Hot-Join command.

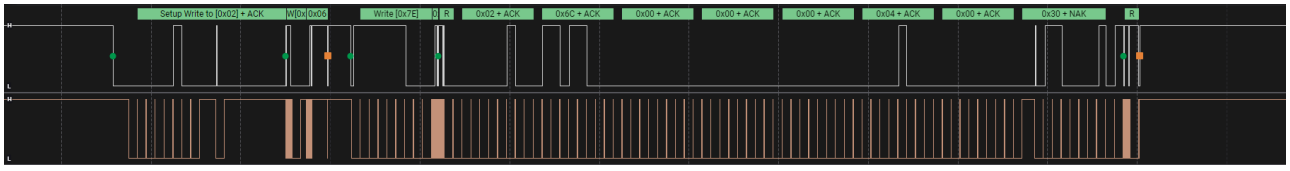

Figure 24. Process timing of Hot-Join command

3 Revision history

Table 9 summarizes the revisions to this document.

Table 9. Revision history

Document ID	Release date	Description
AN14434 v.1.0	20 September 2024	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Microsoft, Azure, and ThreadX — are trademarks of the Microsoft group of companies.

Contents