Application Note

Implementing 1-Wire Enumeration for TMP1826 With TM4C129x Microcontrollers

TEXAS INSTRUMENTS

ABSTRACT

1-Wire bus is used in systems that have low power communication and reduced pin count requirements. This application report describes the 1-Wire communication protocol, available TivaWare™ for C Series APIs for the 1-Wire controller in Tiva™ C Series (TM4C129x) microcontrollers and an example enumeration algorithm using binary tree search. The device used in this example code to enumerate 1-Wire devices is the TMP1826 ±0.3°C accurate digital temperature sensor with 2-Kbit EEPROM.

The source code discussed in this application report can be downloaded here. Additionally, *Implementing Host Controller for TMP1826 Single Wire Temperature Sensor* provides emulated host examples for Tiva C Series microcontrollers that do not have a dedicated 1-Wire controller by using GPIO, UART or SPI peripherals.

Table of Contents

List of Figures

List of Tables

Trademarks

TivaWare™ and Tiva™ are trademarks of Texas Instruments.
All trademarks are the property of their respective owners.

1 Introduction to 1-Wire

1-Wire is a communication bus designed to interface temperature sensors and non-volatile memory devices like TMP1826 with a single wire that supports both communication and power delivery. This system is used for low-speed and low-power communication devices. There are two speed modes available: standard speed and overdrive speed. The achievable data rate with standard speed is typically 8.33 kbit/s, while overdrive mode speed communication can perform up to 90 kbits/s.

This protocol uses a single data line for data transmission from one device to another device. The bus is half duplex so that data can move in both directions, but not at the same time. When required, an extra wire can also be used to power up the devices.

The protocol supports one (single drop) or multiple target devices (multi drop) on the bus. There is also a single controller on the bus that controls the transfer of information on the bus. The controller initiates all transfers on the data line. Transfer of data is only possible between controller and target devices, so data cannot be transferred between devices.

A clock is not required for this protocol as each target device is clocked by an internal oscillator synchronized to the falling edge of the bus. When transferring a byte, the least significant bit is transferred first.

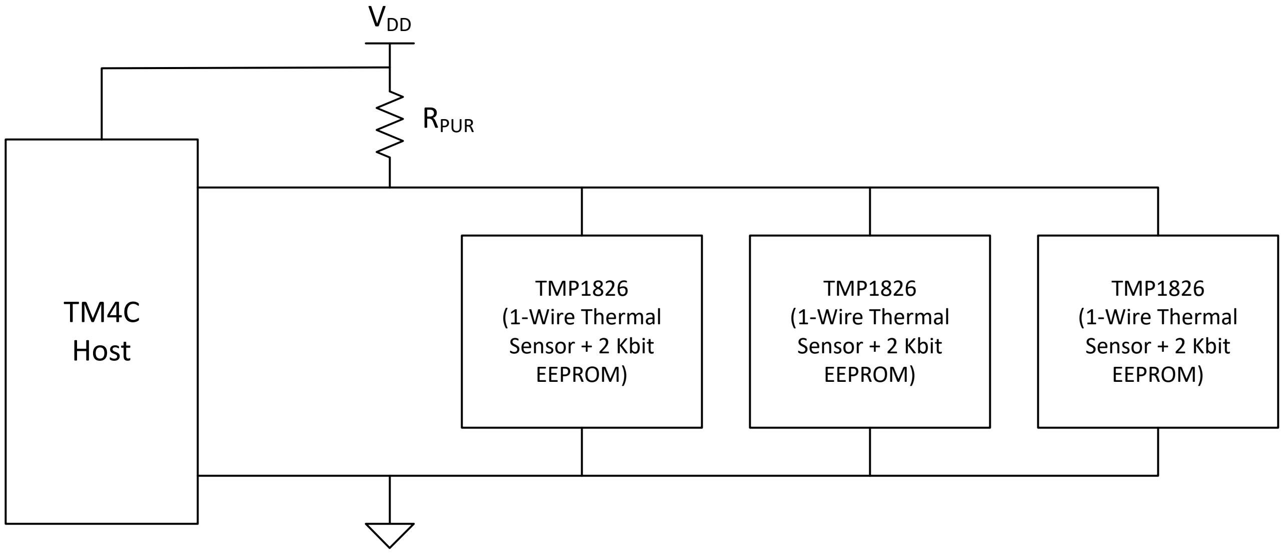
V_{DD}
R_{PUR}
TM4C Host
TMP1826 (1-Wire Thermal Sensor + 2 Kbit EEPROM)
TMP1826 (1-Wire Thermal Sensor + 2 Kbit EEPROM)
TMP1826 (1-Wire Thermal Sensor + 2 Kbit EEPROM)

Figure 1-1. Bus Topology

1.1 Bus Requirements

Every output pin must be open-drain, and a weak pull up must be attached to the signal so that, the bus is driven low if at least one device drives the bus low. This protocol enables the transfer of data between two devices on the bus while the other devices are in the idle state. The strength of the pull up can be decided by the user based on the following factors:

- The device being externally powered may have pull up values in the range of 10K or more if data rate is not required to be high and trace length on the system is not high.
- The device being externally powered may have pull up values less than 1K if the data rate is high especially in the overdrive mode or the trace lengths on the system are longer.
- The device being parasitically powered may require active drivers after being selected so that the end device can have sufficient energy to perform the end operation.

1.2 Powering

Target devices like TMP1826 can be powered in two modes:

- Supply Powered: A dedicated V_{DD} pin on the target device is used to power the bus. This topology is used when the application can support an additional power source near the target devices or additional cable in probe solutions.
- Bus Powered: In this mode the device is powered by the data line. All target devices must have an internal capacitor that stores the energy when the bus is idle and pulled up by the pull-up resistor. The stored energy is then used by the devices during active communication.

2 Functional Description

2.1 Signaling on 1-Wire

The four types of signaling that are possible on the data line are:

- Reset Sequence with Reset Pulse and Answer to Reset (ATR): A reset pulse is used to put all the devices in a known state. Target devices confirm their presence by sending an ATR signal, which is done by holding the line low. The host controller samples the bus, and if the bus reads low, then at least one target device is present.

Table 2-1. Reset Signaling Description and Implementation

Operation	Description	Implementation
Reset	Reset the 1-Wire bus target devices and prepare them for a command.	Drive the bus low for 480 µs to reset all the devices. The host then samples the bus for the next 240 µs while the target Answer to Reset (ATR).

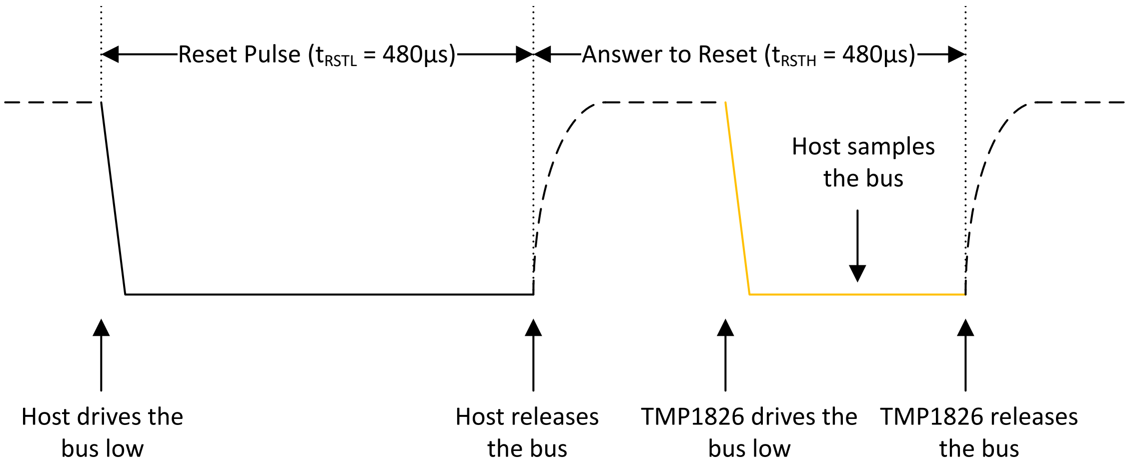
Reset Pulse (t_{RSTL} = 480µs)
Answer to Reset (t_{RSTH} = 480µs)
Host samples the bus
Host drives the bus low
Host releases the bus
TMP1826 drives the bus low
TMP1826 releases the bus

Figure 2-1. Reset Sequence Bus Timing When There is at Least 1 Device on the Bus

- Write logic 0 onto the bus

Table 2-2. Write Logic 0 Bit Signaling Description and Implementation

Operation	Description	Implementation
Write logic 0	Send 0 bit to the 1-Wire target devices	Drive the bus low for 60 µs

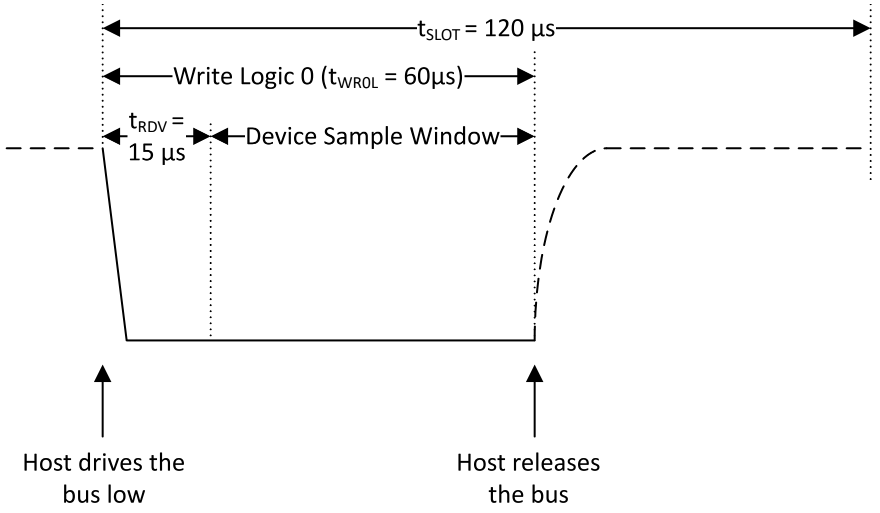

Figure 2-2. Write Logic 0 Bus Timing

- Write logic 1 onto the bus

Table 2-3. Write Logic 1 Signaling Description and Implementation

Operation	Description	Implementation
Write 1 bit	Send 1 bit to the 1-Wire target devices	Drive the bus low for < 15 µs. Typical times are about 6 µs. Release the bus until 60 µs after the falling edge.

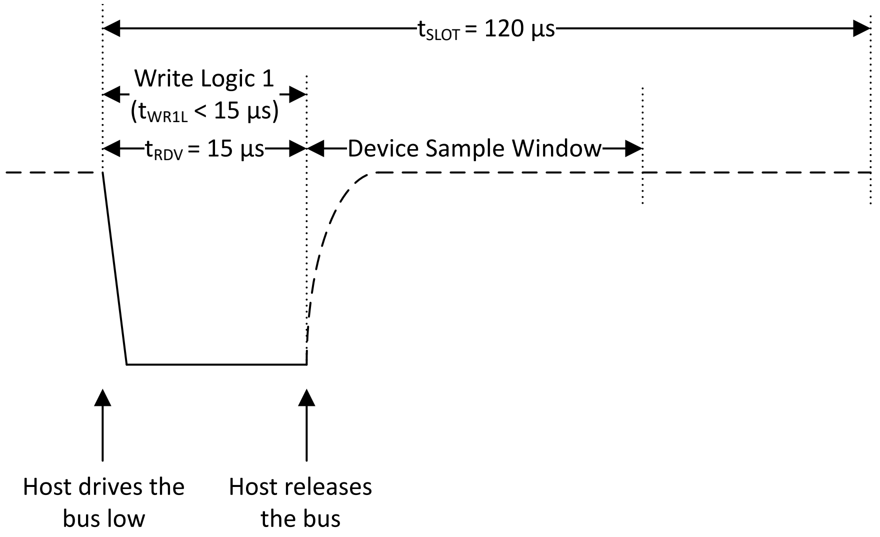

Figure 2-3. Write Logic 1 Bus Timing

- Read bit: Reads one bit from the target devices. Read bit signaling is similar to write “1” signaling, except that the host controller reads instead of writes.

Table 2-4. Read Bit Signaling Description and Implementation

Operation	Description	Implementation
Read bit	Read a bit from the 1-Wire target device	Drive the bus low from 1 µs to 15 µs. Sample the bus at 15 µs after the falling edge to read the bit from the target device.

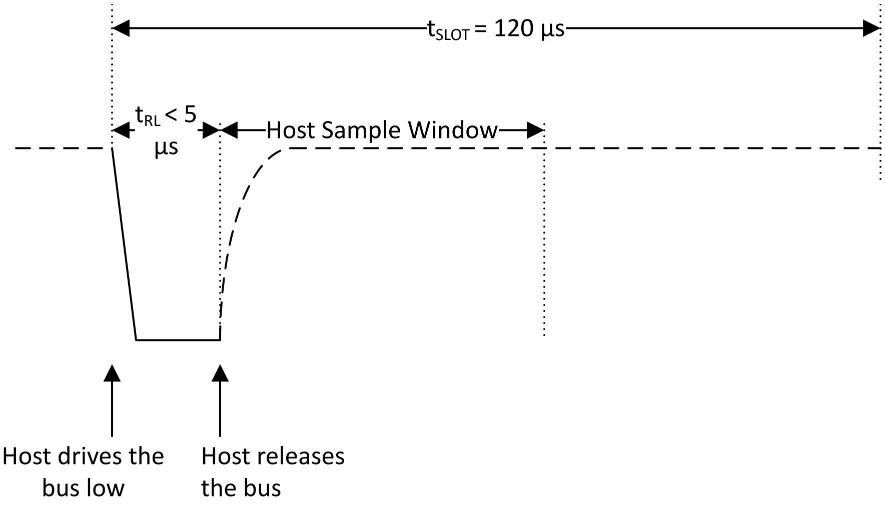

Figure 2-4. Read Logic 1 Bus Timing

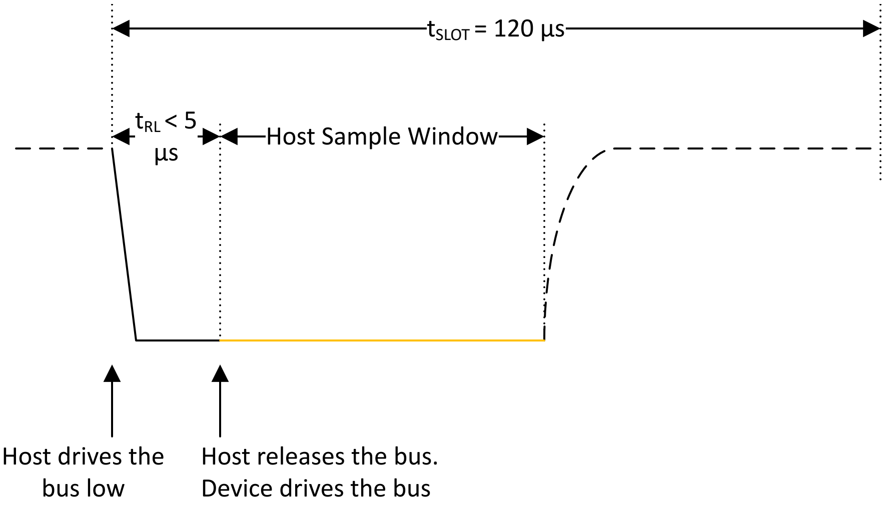

Figure 2-5. Read Logic 0 Bus Timing

2.2 Address Format of the 1-Wire Device

1-Wire target devices from Texas Instuments have a unique 64-bit address stored in it, which is also called the device address. The least significant 8 bits of the address give the family code of the device. The next least 48 bits give the serial number of the device. The most significant 8 bits give the CRC generated from the least 56 bits.

Table 2-5. Address Format of the 1-Wire Target Device

	64-Bit Device Address	
8-Bit CRC	48-Bit Serial Number	8-Bit Family Code
63:56 Bits	55:8 Bits	7:0 Bits

Note

The CRC is used to check if the data is received correctly. Tiva C Series devices do not implement the CRC in hardware, so a software implementation is required.

2.3 Typical Communication Flow on the 1-Wire Bus

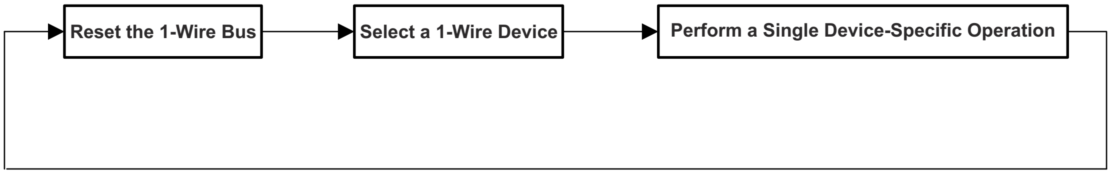

Figure 2-6. Typical Communication Flow on a 1-Wire Bus

- Start with the Reset Sequence
- If the host must determine which target devices are present on the bus, it should perform a search to detect the 64-bit device address of the devices.
- Before performing an operation on a device, the device must be configured and/or selected using the ROM commands. Some of the available functional ROM commands are:
  - Read Address [0x33]: Only used when there is a single device on the bus. This command reads the 64-bit device address of the only device present on the bus.
  - Match Address [0x55]: This command followed by a 64-bit device address selects the device with the matching address. All other devices wait until the next reset pulse.
  - Search Address [0xF0]: This command is required to obtain the 64-bit device address of multiple devices, and it informs devices that a search is going to be conducted by the host. The Search is then conducted by reading a bit and its complement of the ROM numbers from the devices and sending an appropriate bit back. For more details, see the Section 4. Target devices that have the same bit as the one sent by the host remain active while others wait for the next reset
  - Skip Address [0xCC]: Devices can be addressed without the host knowing the 64-bit device address. This command is helpful when giving a common command to all the devices.
  - Overdrive Skip Address [0x3C]: This command is used only in single drop. This command is the same as the Skip Address command except that only devices that can run in overdrive mode remain active and go into overdrive mode. Devices that cannot run in overdrive mode wait for the next reset.
  - Overdrive Match Address [0x69]: This command is the same as the Match Address command except that the device is only matched if it can run in overdrive mode. All other devices wait for the next reset.
- After selecting the required devices, a device-specific command can be issued to perform the required operation.
- Typically after each operation, a reset pulse is issued.

3 Functions Available in TivaWare for C Series for 1-Wire Module

Note

ui32Base should contain the base address of the 1-Wire module.

OneWireBusReset(uint32_t ui32Base); This function issues a reset on the 1-Wire bus; it does not wait for the completion of the reset.

OneWireBusStatus(uint32_t ui32Base); This function retrieves the bus condition status, which is used to determine if the bus is ready to perform an operation. This status is busy if the host is performing any operation or idle if the host is not performing an operation.

OneWireDataGet(uint32_t u3i2Base, uint32_t *pui32Data);This function waits for the transaction, if any, to complete and retrieves data from the 1-Wire interface. The data is stored in the address given by pui32Data.

OneWireDataGetNonBlocking(uint32_t ui32Base, uint32_t *pui32Data); This function retrieves data from the 1-Wire interface. If there is an active transaction, then 0xffffffff is returned. The data is stored in the address given by pui32Data.

OneWireInit(uint32_t ui32Base, uint32_t ui32InitFlags); This function initializes the 1-Wire module. The ui32InitFlags parameter contains the initialization flags.

OneWireIntClear(uint32_t ui32Base, uint32_t ui32IntFlags); This function clears the required interrupt sources in the 1-Wire module.

OneWireIntDisable(uint32_t ui32Base, uint32_t ui32IntFlags); This function disables the required interrupt sources in the 1-Wire module.

OneWireIntEnable(uint32_t ui32Base, uint32_t ui32IntFlags); This function enables the required interrupt sources in the 1-Wire module.

OneWireIntRegister(uint32_t ui32Base, void(*pfnHandler)(void)); This function registers an interrupt handler for the 1-Wire module.

OneWireIntUnregister(uint32_t ui32Base); This function unregisters an interrupt handler for the 1-Wire module.

OneWireIntStatus(uint32_t ui32Base, bool bMasked); This function gets the current interrupt status. If bMasked is true, then the masked interrupt status is obtained. If bMasked is false then the raw interrupt status is obtained.

OneWireTransaction(uint32_t ui32Base, uint32_t ui32OpFlags,uint32_t ui32Data, uint32_t ui32BitCnt); This function is used to perform a 1-Wire protocol transfer on the bus. ui32BitCnt is used to configure the number of bits to be sent or received. Written data is specified by ui32Data. The ui32OpFlags parameter is used to define which operation (reset, read, write) is to be performed.

Note

More information about the APIs can be found in the (*TivaWare™ Peripheral Driver Library User's Guide*).

4 Enumeration

When multiple devices are present on the bus, the host must know the 64-bit device address to match the required devices. Enumeration is performed to obtain the 64-bit address of the devices on the bus and therefore what classes of devices are available.

The legacy mode for search address operation described in Section 4.1 is performed by executing two steps, iteratively:

1. Read two bits from the devices (the actual bit value and its complement)
2. Write the appropriate bit that defines the search path in the enumeration algorithm. Devices with the corresponding bit equal to the written bit remain active, while the rest go to the idle state and wait for the next reset command.

This cycle is done for all 64 bits and the 64-bit device address is assembled at the end of the iteration.

TMP1826 supports fast search operation which greatly simplifies the 64-bit device address and is described in Section 4.2

4.1 Legacy Search Algorithm

The Search algorithm uses a binary search tree. At every node, the algorithm can take the path dictated by either a "0" or "1." The relationship between the two bits obtained in step 1 and the path to be taken in step 2 are given in Table 4-1.

Table 4-1. Search Algorithm

Actual Bit Read Value	Complement Bit Read Value	Conclusions	Path to be taken
0	0	Multiple devices have a corresponding 0 bit and a corresponding 1 bit	This is a conflict situation and requires a decision on which path to take
1	0	Only one device has a 1 in the corresponding bit location	The 1 path is taken
0	1	Only one device has a 0 in the corresponding bit location	The 0 path is taken
1	1	No devices are on the bus	End the search

The only time that a decision must be made is when there is a conflict. In the other three cases, the path to be taken is already defined. Figure 4-1 shows the algorithm flow for a 4-bit search. The following variables in software are key to the search.

- The variable *i32LastConflictZeroBitNumber* stores the iteration number of the last conflict node where the path taken is “0” while finding the current ROM number.
- The variable *li32ConflictBitNumber* stores the value of last_conflict_zero for the last ROM number.
- The variable *ui32BitNumber* gives the position of the bit under consideration in the ROM number.

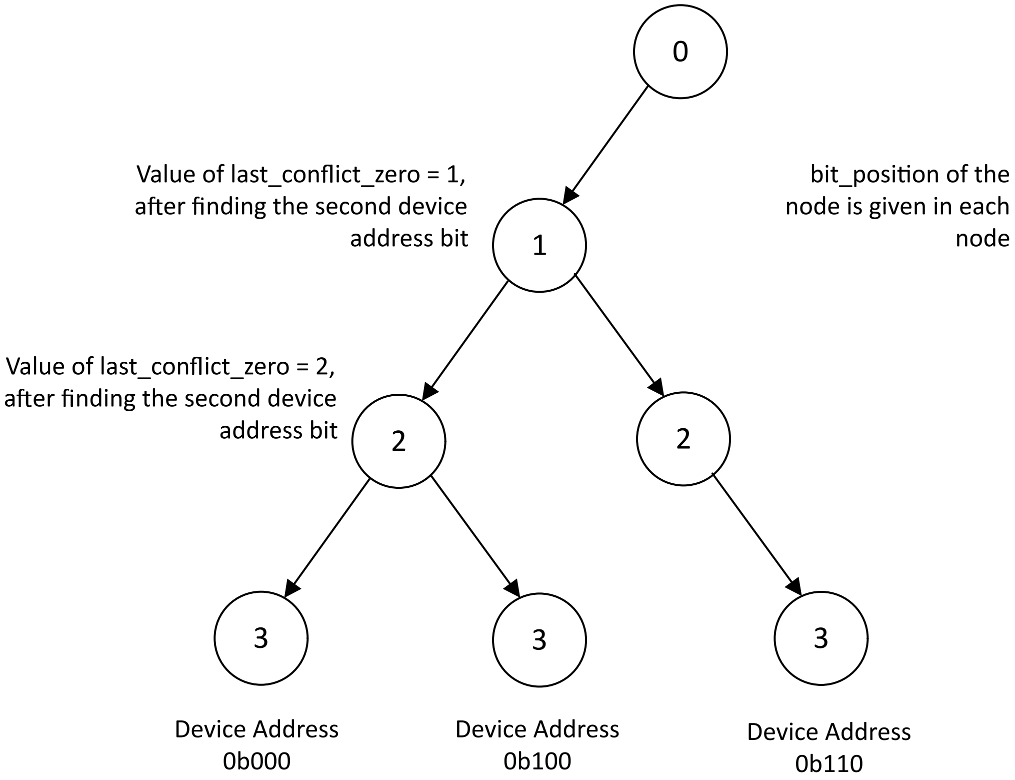
0
Value of last_conflict_zero = 1, after finding the second device address bit
1
bit_position of the node is given in each node
Value of last_conflict_zero = 2, after finding the second device address bit
2
2
3
3
3
Device Address 0b000
Device Address 0b100
Device Address 0b110

Figure 4-1. Enumeration using binary search tree

4.1.1 Steps of the 3-Bit Search Algorithm

1. Reset the bus and look for ATR responses. End the process if there are no devices on the bus.
2. Send the Search ROM command if an ATR response is received.
3. Read a bit from the devices on the bus.
4. Read the complement of the bit in step 3 from the devices.
5. Check if both the bits are 1. If ‘yes,’ end the process. If ‘no’, continue.
6. Check if the first read bit is a 0 and the second read bit is a ‘1.’ If ‘yes,’ write 0 onto the bus and go to step 14. If ‘no’, continue.
7. Check if the first read bit is a 1 and the second read bit is a zero. If ‘yes’, go to step 9. If ‘no’, continue.
8. Check whether or not *ui32BitNumber* is equal to *i32ConflictBitNumber*. If ‘yes’, continue. If ‘no’, go to step 10
9. Write 1 onto the bus. Go to step 14.
10. Check whether or not *ui32BitNumber* is less than *i32ConflictBitNumber*. If ‘yes’, continue. If ‘no’, go to step 12.
11. Check whether or not the bit in the *ui32BitNumber* of the last ROM number is equal to 1. If ‘yes’, go to step 9. If ‘no’, continue.
12. Write 0 onto the bus.
13. Update the value of *i32LastConflictZeroBitNumber* with the *ui32BitNumber*.
14. Check whether or not *ui32BitNumber* is equal to 63. If ‘yes’, go to step 1. If ‘no’, go to step 3.

4.2 Fast Search Algorithm

The fast search algorithm uses the TMP1826 devices to arbitrate. The host issues a SKIP ADDRESS command and writes the configuration register-2 bits for ARB_MODE as 11b. After performing the last step it issues a SEARCH ADDRESS command and reads 1-bit of the device address. All devices on the bus transmit their address as either logic 1 or 0 and monitor the SDQ line to read back what they have transmitted. Since the logic 0 is a dominant value, any device that sent logic 1 will lose the bus and no longer participate in the current iteration of the address search operation.

The host then reads the next 1-bit of the device address from the devices that have won the bus arbitration. This process continues till only one device successfully sends its 64-bit device address at which point the device sets a success flag in its status register. The host then sends another SEARCH ADDRESS command and all the devices except the one which have been enumerated rejoin the process of address enumeration.

The fast search algorithm compared to the legacy search algorithm from older 1-wire devices is 3 times faster and saves the memory and processing cost of implementing a binary tree search. For more details on the method described and re-enumeration, see the *TMP1826 Single-Wire, ±0.3°C Accurate Temperature Sensor With 2-Kbit EEPROM Data Sheet*.

5 Summary

This application report provides an overview of the 1-Wire protocol and presents an example of how the host identifies the TMP1826 on the bus using binary tree search.

6 References

- Texas Instruments: *TMP1826 Single-Wire, ±0.3°C Accurate Temperature Sensor With 2-Kbit EEPROM*
- Texas Instruments: *TivaWare™ Peripheral Driver Library User's Guide*
- Texas Instruments: *Implementing Host Controller for TMP1826 Single Wire Temperature Sensor*
- TivaWare™ for C Series Software

7 Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Revision C (January 2018) to Revision D (June 2022) **Page**

- Replaced generic 1-wire device with TMP1826 throughout the document. 2
- Updated the numbering format for tables, figures and cross-references throughout the document. 2

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265