Application Note 192
Using the DS2480B Serial 1-Wire Line Driver

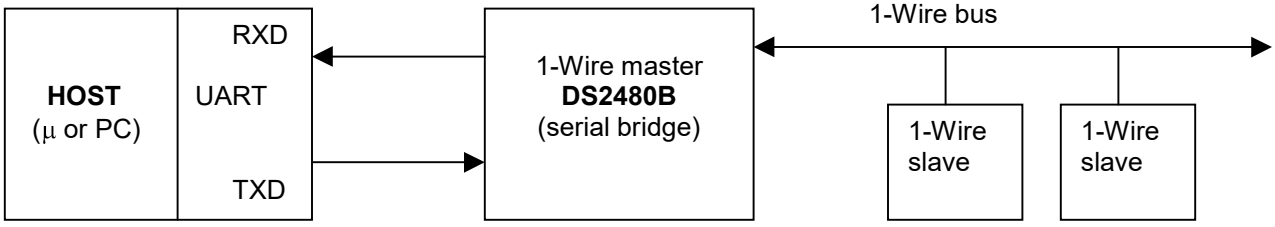
INTRODUCTION

The 1-Wire® communication protocol can be generated with an IO pin on a microprocessor; however, care must be taken to provide the correct timing and proper slew rates to create a reliable 1-Wire network. Improper timing sent by the 1-Wire master may cause communication with 1-Wire slave devices to be intermittent or fail altogether. Uncontrolled slew rates can severely limit the length of a network and create sporadic behavior. If a serial communication UART is available, using a serial-to-1-Wire bridge (DS2480B) will eliminate these problems.

The DS2480B is a serial bridge to the 1-Wire network protocol. This bridge allows any host with a very modest serial communication UART to generate properly timed and slew controlled 1-Wire waveforms. The DS2480B receives escaped commands and data, performs 1-Wire operations, and returns the result back to the host. See Figure 1 for a simplified diagram of the DS2480B configuration. Implementation of this protocol and navigating the available DS2480B commands can be time consuming and confusing. This guide identifies common 1-Wire operations and explains the construction of input serial packets and interpretation of output serial packets for the DS2480B.

This document is designed to complement the DS2480B data sheet, but not replace it. The data sheet can be found on https://datasheets.maximintegrated.com/en/ds/DS2480B.pdf

DS2480B USAGE (SIMPLIFIED) Figure 1



The minimum host UART that will work with this bridge must support 8-bit, non-parity, 9600 baud (bits per second) communication. Faster data rates up to 115200 baud can be negotiated but the bridge starts at 9600 baud when powering up. Electrical considerations such as RS232 are addressed in DS2480B data sheet.

THE 1-WIRE INTERFACE

The DS2480B is only useful if all of its commands and modes can be translated into a 1-Wire communication interface that applications can use and build upon. There are a few basic 1-Wire functions that an application must have in order to do any 1-Wire operation. This first operation resets all of the 1-Wire slaves on the bus readying them for a command from the 1-Wire master. The second writes a bit from the 1-Wire master to the slaves and the third reads a bit from the 1-Wire slaves. Since the 1-Wire master must start all 1-Wire bit communication, a 'read' is technically a 'write' of a one bit with the result sampled. Almost all other 1-Wire operations can be constructed from these three operations. For example, a byte written to the 1-Wire bus is just eight single bit writes. The 1-Wire Search Algorithm (See Application Note 187 at https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/187.html) can also be constructed using these primitives. This is not necessarily the most efficient implementation method, however. The DS2480B incorporates a search accelerator mode that greatly reduces the serial

communication required to do a search. It is also more efficient to bundle groups of bit communication into bytes and even blocks of bytes. Whenever possible, an application should use the largest grouping of commands (biggest packets) for maximum efficiency.

Table 1 is a minimal interface of efficient 1-Wire functions. The operation name is provided as a label to the particular operation and will be used through the remainder of this document.

BASIC 1-WIRE OPERATIONS Table 1

Operation	**Description**
OWReset	Send the 1-Wire reset stimulus and check for 1-Wire slave device presence pulses.
OWWriteBit / OWReadBit	Send or receive a single bit of data to the 1-Wire bus.
OWWriteByte / OWReadByte	Send or receive a single byte of data to the 1-Wire bus.
OWBlock	Send and receive multiple bytes of data to and from the 1-Wire bus.
OWSearch	Perform the 1-Wire Search Algorithm (see Application Note 187).

There are also extended 1-Wire functions that are not covered in the basic operations. Some 1-Wire slave devices can operate at two different communication speeds: standard and overdrive. All devices at least support standard. Overdrive is approximately 10 times faster than standard. The DS2480B supports both 1-Wire speeds.

1-Wire devices normally derive some or all of their operating energy from the 1-Wire bus. However some devices require additional power delivery at a particular place in the protocol. For example, a device may need to do a temperature conversion or compute an SHA-1 hash. This power is supplied by enabling a stronger pullup on the 1-Wire bus. Normal communication cannot take place during this power delivery. The DS2480B has several advanced features to provide power delivery.

EPROM (one-time-programmable) 1-Wire memory devices require a special 12V pulse when writing. If the DS2480B has 12V available then it can be told to deliver a pulse onto the 1-Wire bus for EPROM programming.

Table 2 lists the extended 1-Wire operations for 1-Wire speed, power delivery, and programming pulse.

EXTENDED 1-WIRE OPERATIONS Table 2

Operation	**Description**
OWSpeed	Set the 1-Wire communication speed. The choice is standard or overdrive speed. Note that this only changes the communication speed of the 1-Wire master; the 1-Wire slave device must be instructed to make the switch when going from normal to overdrive. The 1-Wire slave will always revert to standard speed when it encounters a standard speed 1-Wire reset.
OWLevel	Set the 1-Wire power level (normal or power delivery).
OWProgramPulse	Sends a timed 12V programming pulse for EPROM 1-Wire device writing.
OWReadBitPower	Read a single bit of data from the 1-Wire bus and optionally apply power-delivery immediately after the bit is complete.
OWWriteBytePower	Send a single byte of data to the 1-Wire bus and apply power-delivery immediately after the byte is complete.

This document presents an efficient implementation of the basic and extended 1-Wire operations using the DS2480B. These operations provide a complete foundation to perform all functions for current and future 1-Wire devices. Abstracting the 1-Wire operations in this fashion leads to 1-Wire applications that are independent of the 1-Wire master type (see Example 1). Note that this implementation is not the only one possible and does not necessarily utilize all of the DS2480B's features. More 1-Wire usage examples are presented at the end of this document.

READ MEMORY PSEUDO CODE Example 1

```
trans_block – temporary transmit buffer, values expressed in hexadecimal notation

// reset the 1-Wire bus
If OWReset = TRUE

   // sent the MATCH ROM sequence for the device to read, ROM is R0...R7
   trans_block = 55,R0,R1,R2,R3,R4,R5,R6,R7
   OWBlock(trans_block)

   // send the Read Memory command, address (0),
   // and 32 read bytes for the page of data
   trans_block = F0,00,00,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,
                 FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,FF,  FF,FF,FF,FF
   OWBlock(tran_block)

   // page 0 data is now in last 32 bytes of trans_block
   ...
Else
   // no device present
   ...
EndIF
```

HOST CONFIGURATION

The host of the DS2480B must have a UART that can at least support 8-bit, non-parity, 9600 baud communication. Configuration of the host UART is platform specific so is not covered by this document. It must, however, provide standard interface operations included reading and writing, flushing any pending reads/writes, issuing a break, and optionally changing the baud rate. Table 3 provides a list of operation terms to describe an interface to a generic host UART. These terms will be used to describe operations done to the UART.

REQUIRED HOST UART OPERATIONS Table 3

Operation	**Description**
Break	Sends a BREAK on the communication port for at least 2ms.
Flush	Allows any pending write operations to complete and clear input (read) buffer.
Read	Read a specified number of bytes from the serial port. Provide a sufficiently long timeout to ensure that all bytes are received under normal conditions.
Write	Write a specified number of bytes to the serial port. Return after all of the bytes have been written.
SetBaud	Changes the serial BAUD rate to the rate specified. (Optional if overdrive needed.)
Delay	Delays at least the specified number of milliseconds.

A ‘C’ code implementation of this application note using the Microsoft Windows 32-bit operating system RS232 serial port with a DS9097U adapter as the host can be downloaded from the following link:
http://files.maximintegrated.com/sia_bu/public/an192.zip

This ‘C’ code implementation is a simplified version of the one provided in the 1-Wire Public Domain kit. The 1-Wire Public Domain kit also contains device specific modules and examples and can be found at the following link:
https://www.maximintegrated.com/en/design/design-tools/applications-software/product-software/ibutton/software/1wire/wirekit.html

DS2480B CONFIGURATION

Before any 1-Wire operations can be attempted, the host must setup and synchronize with the DS2480B Serial 1-Wire line driver. This setup and synchronization procedure is also done if a communication problem is ever detected between the host and the bridge. The DS2480B requires 9600 baud during setup. After setup the baud rate can be negotiated up to 115200 baud. Care must be taken, however, since the DS2480B only has a one byte input buffer. The provided 1-Wire command must be able to complete before the next command is shifted in. See Table 7 in the DS2480B data sheet to see what commands will work at what baud rates.

DS2480B_Detect

Since the DS2480B does not have a crystal it must tune its time-base by sampling the serial communication sent by the host. This setup sequence is initiated by resetting the DS2480B and then sending a predefined timing byte. Resetting the device will result in all of the 1-Wire configuration parameters being reset to their default state. For good performance on small to medium length 1-Wire networks, it is recommended that the DS2480B be used in ‘flex’ mode when doing standard speed communication. The 1-Wire configuration parameters are used to shape the 1-Wire signal in flex mode. Consequently, whenever the DS2480B is reset, the configuration parameters need to be reloaded. The desired flex settings are PDSRC=1.37V/µs, W1LD=10µs, DSO/W0RT=8µs. This reset and configuration sequence is combined into on operation called *DS2480B_Detect*.

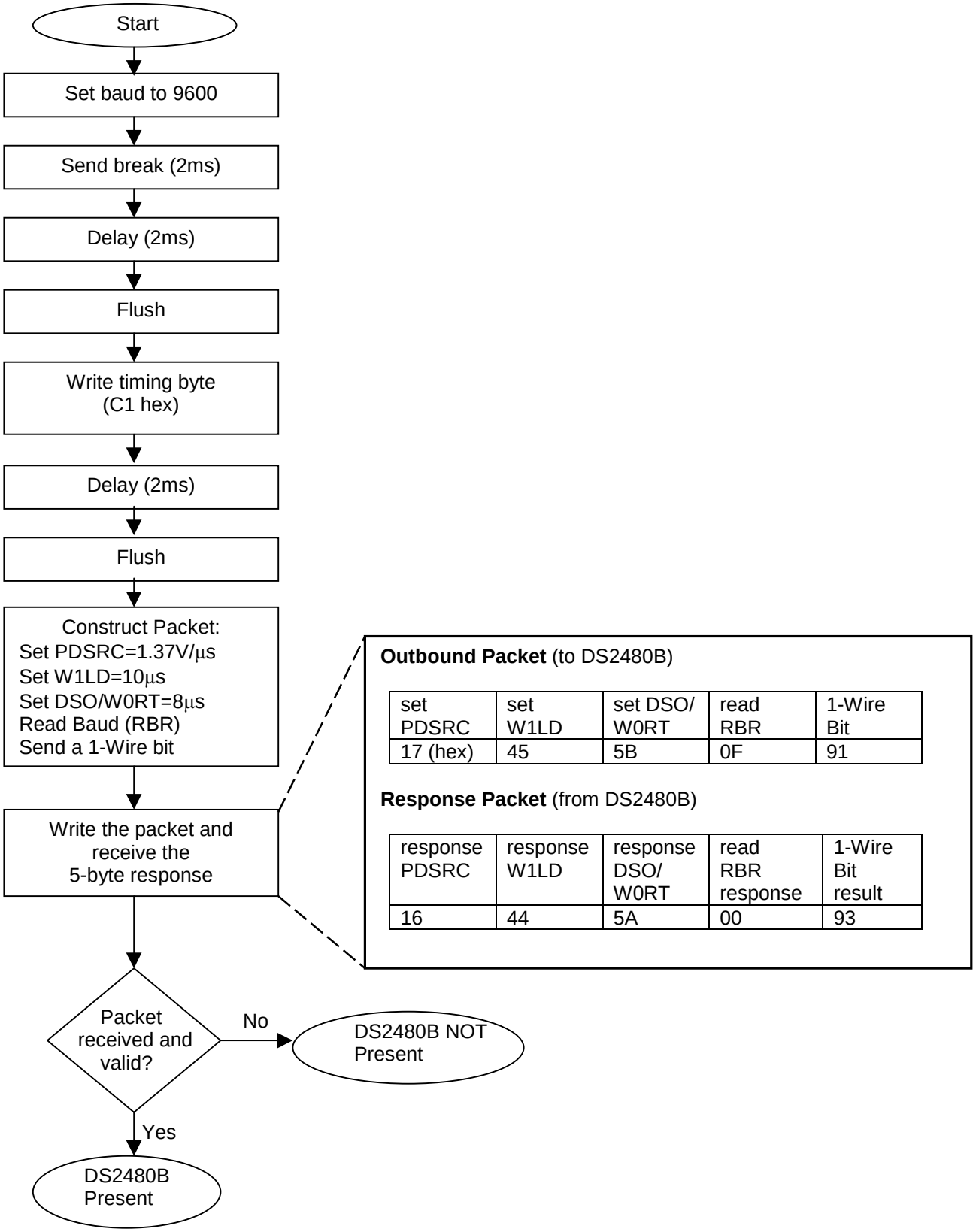
The DS2480B is reset if it detects a space in the stop-bit position. The easiest way to generate this is with a serial break longer than a 9600 baud 8-bit word. If break is not available on the host UART then switching to a slower baud rate and sending a zero byte can simulate a break. Switching to space parity or changing to a 9-bit word length with a zero in the most significant bit can also simulate a break.

Some of the delay values in the configuration sequence (see Figure 2) are arbitrarily large to accommodate most UARTS. These values can be reduced.

The read baud rate register and write 1-Wire bit at the end of the setup sequence is designed to measure the correct functioning of the DS2480B setup. If either one of those operations returns an invalid response then the setup is deemed unsuccessful.

Note that this implementation does not check for the unsolicited presence pulse notification byte from the DS2480B. This may cause any of the 1-Wire operations to get an improperly formatted response byte leading to the call to the *DS2480B_Detect* function. Since any 1-Wire application that is used in an intermittent contact environment that would produce these unsolicited presence pulse notifications must already incorporate retries, this does not present a problem.

DS2480B_DETECT FLOW Figure 2



Outbound Packet (to DS2480B)

set PDSRC	set W1LD	set DSO/ W0RT	read RBR	1-Wire Bit
17 (hex)	45	5B	0F	91

Response Packet (from DS2480B)

response PDSRC	response W1LD	response DSO/ W0RT	read RBR response	1-Wire Bit result
16	44	5A	00	93

DS2480B_ChangeBaud

To change the communication speed between the HOST and the DS2480B, the RBR (RS232 Baud Rate) register must be written. The DS2480B immediately responds with the set baud rate response at the new baud rate but it is likely to be missed by the HOST. Consequently, the recommended flow as seen in Figure 3 has the response byte flushed and ignored.

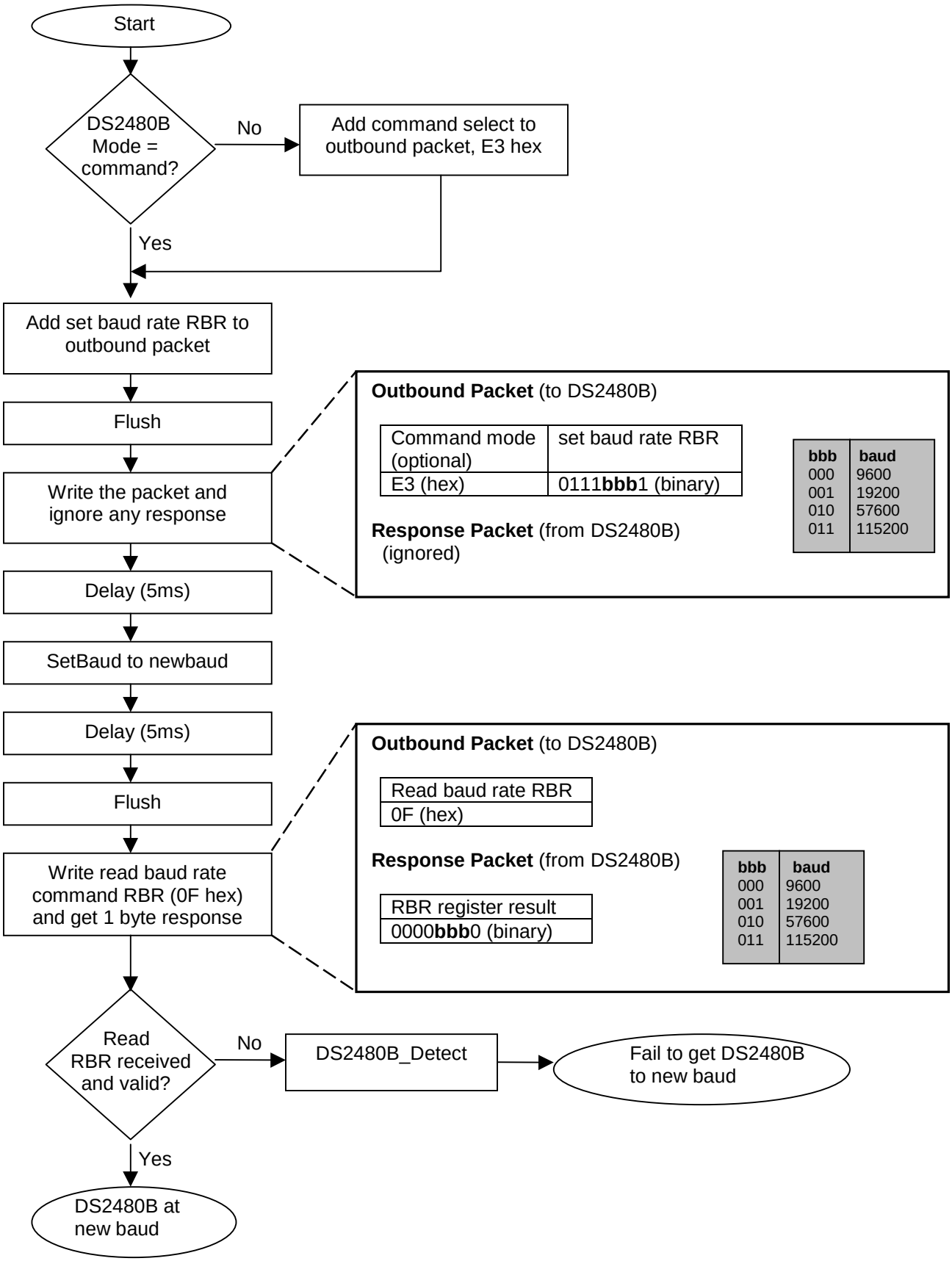
After both the host and the DS2480B have switched baud rates, the baud rate register is read back to verify completion. If the DS2480B is not at the correct baud rate, the read back will fail and the setup initialization sequence *DS2480B_Detect* is called.

The DS2480B can operate on four different baud rates: 9600, 19200, 57600, and 115200. Since the DS2480B has only a one-byte buffer, the command sent must be complete before the next command arrives. Figure 7 in the data sheet shows what operations can be performed at what baud rates without danger of overwriting commands. It is recommended that the 1-Wire reset result should always be received before proceeding to the next command so do not include it in a continuous byte stream. This implementation splits the 1-Wire reset off into its own operation *OWReset* so this is not an issue. Similarly, the single bit and single byte operations are also split into their own operations: *OWReadBit, OWWriteBit, OWReadByte,* and *OWWriteByte*. Since they are not streamed with other commands, the maximum baud rate can be used. Also note that this implementation only uses flex mode with extended bit timing when doing standard speed 1-Wire communication which affects the allowed baud rates. See Table 4 for the baud rate recommendations. For simplicity, this implementation will use only two baud rates: 9600 baud for standard speed (flex) operations and 115200 baud for non-search overdrive operations.

MAXIMUM STREAMING BAUD RATES Table 4

Function	**Standard (Flex)**	**Overdrive**
Search (OWSearch)	9600 (baud)	57600
Command (all non 1-Wire operations)	115200	115200
Data (OWBlock)	9600	115200

DS2480B_CHANGEBAUD FLOW Figure 3



1-WIRE OPERATIONS

The basic and extended 1-Wire operations create a common 1-Wire interface that facilitates any operation on any 1-Wire device.

The implementation for each of these 1-Wire operations has some common features. The commands and data are grouped together whenever possible to reduce the number of packets exchanged with the DS2480B. The current mode of the DS2480B is kept as state so a packet may start with a mode changing command. If the response packet is not the correct length or is in an improper format, the *DS2480B_Detect* sequence is called.

The only change in the serial communication rate between the host and the DS2480B is done in *OWSpeed* when changing the 1-Wire communication speed. As it is implemented, *OWSearch* cannot be run in overdrive mode. A simple check could be added to reduce the baud rate to 57600 baud when doing a search at the faster communication rate.

Every 1-Wire operation should first make sure that the current level pullup is normal. So each flow starts with an implicit call to *OWLevel(normal)*.

OWReset

The OWReset operation instructs the DS2480B to send a reset pulse to the 1-Wire and sample to detect the presence pulses from 1-Wire slaves on the bus. While the primary purpose of this instruction is to perform this reset operation, it also returns other useful information. It also provides a three-bit field that indicates the version of the chip. The version field will be constant with all DS2480Bs, however it can be used to detect the predecessor to this bridge, the DS2480. This implementation is compatible with the DS2480. Also, masking off this field will make the host software or firmware at least partially compatible to future bridge versions.

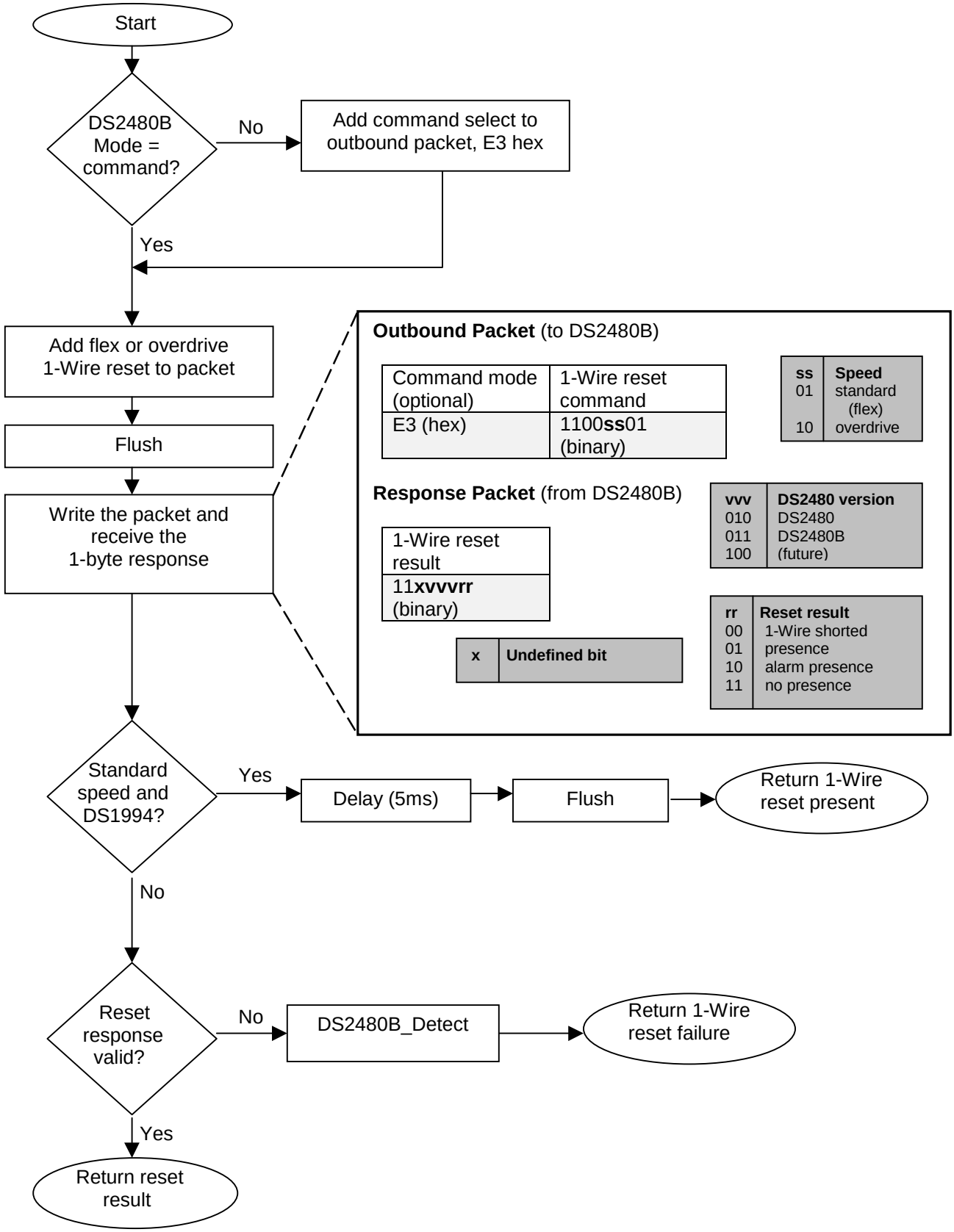
The 1-Wire reset command operation takes in the communication speed. Note that when communicating in standard 1-Wire speed, the DS2480B flex mode is used in this implementation.

The time that this operation takes to complete depends on whether there is an alarm presence. This is the main reason why this operation is not grouped in packets with other 1-Wire operations. Note the extra 5ms delay and flush if a DS1994* or DS2404* device could be on the 1-Wire bus. The DS2480B does not handle all of the 1-Wire reset alarm types from these devices and communication must be delayed until they are complete.

See Figure 4 for the flow of this operation.

*The DS1994 and DS2404 are no longer recommended for new designs.

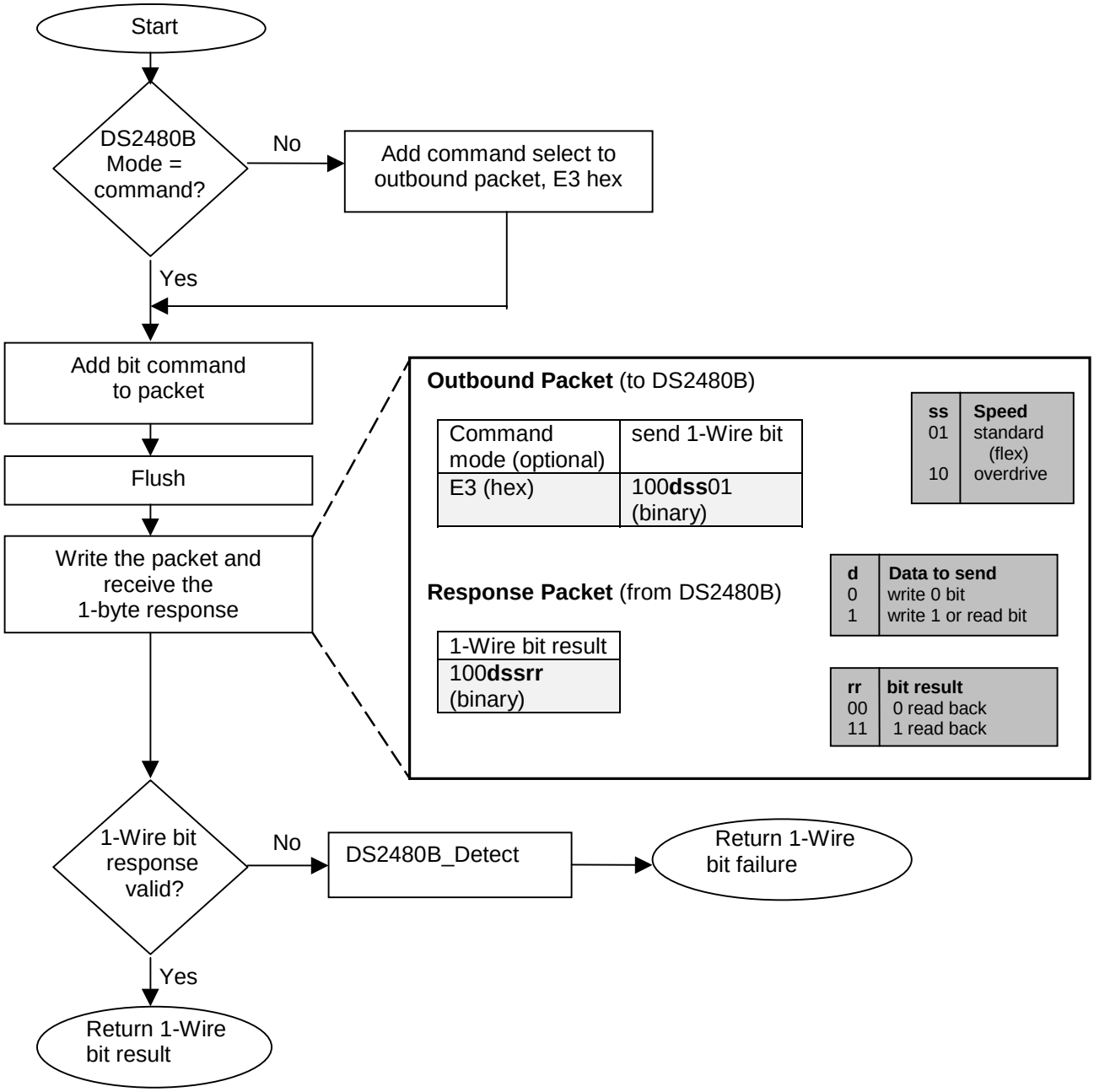
OWRESET FLOW Figure 4



OWWriteBit / OWReadBit

Performing a single bit operation on the 1-Wire is unusual but is included here for completeness. When the protocol indicates a write bit then the value is just written to the 1-Wire as seen in Figure 5. If a read is required then a write-one is done and the result sampled is the read result.

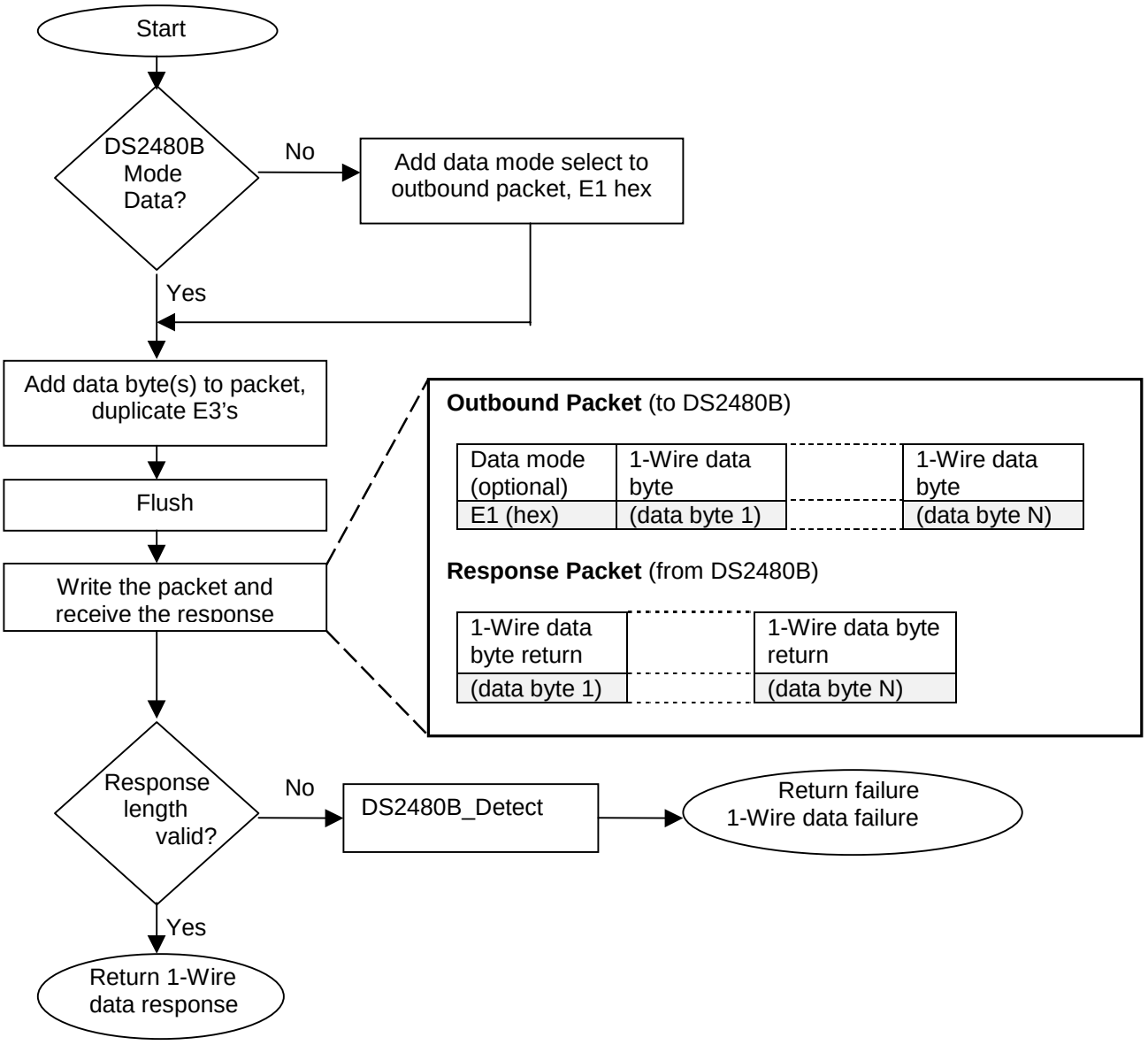
OWWRITEBIT/OWREADBIT FLOW Figure 5



OWWriteByte / OWReadByte / OWBlock

The single and multiple byte operations are very similar. The DS2480B must first be put into data mode. Like the single bit commands, when the protocol indicates a write byte then the data is just written. A read byte is done by writing a FF hex (all ones) and then the sampled data is the read result. A block operation is a group of single byte operations that may be a mixture of reads and writes. The read positions must be pre-filled with FF's hex. Care must be taken to always duplicate data bytes that are the same as the switch to command mode command (E3 hex). This instructs the DS2480B to treat it as data and not the switch to command mode.

OWWRITEBYTE/OWREADBYTE/OWBLOCK FLOW Figure 6



OWSearch

The search algorithm is a binary tree search where branches are followed until a device ROM number, or leaf, is found. Subsequent searches then take the other branch paths until all of the leaves present are discovered.

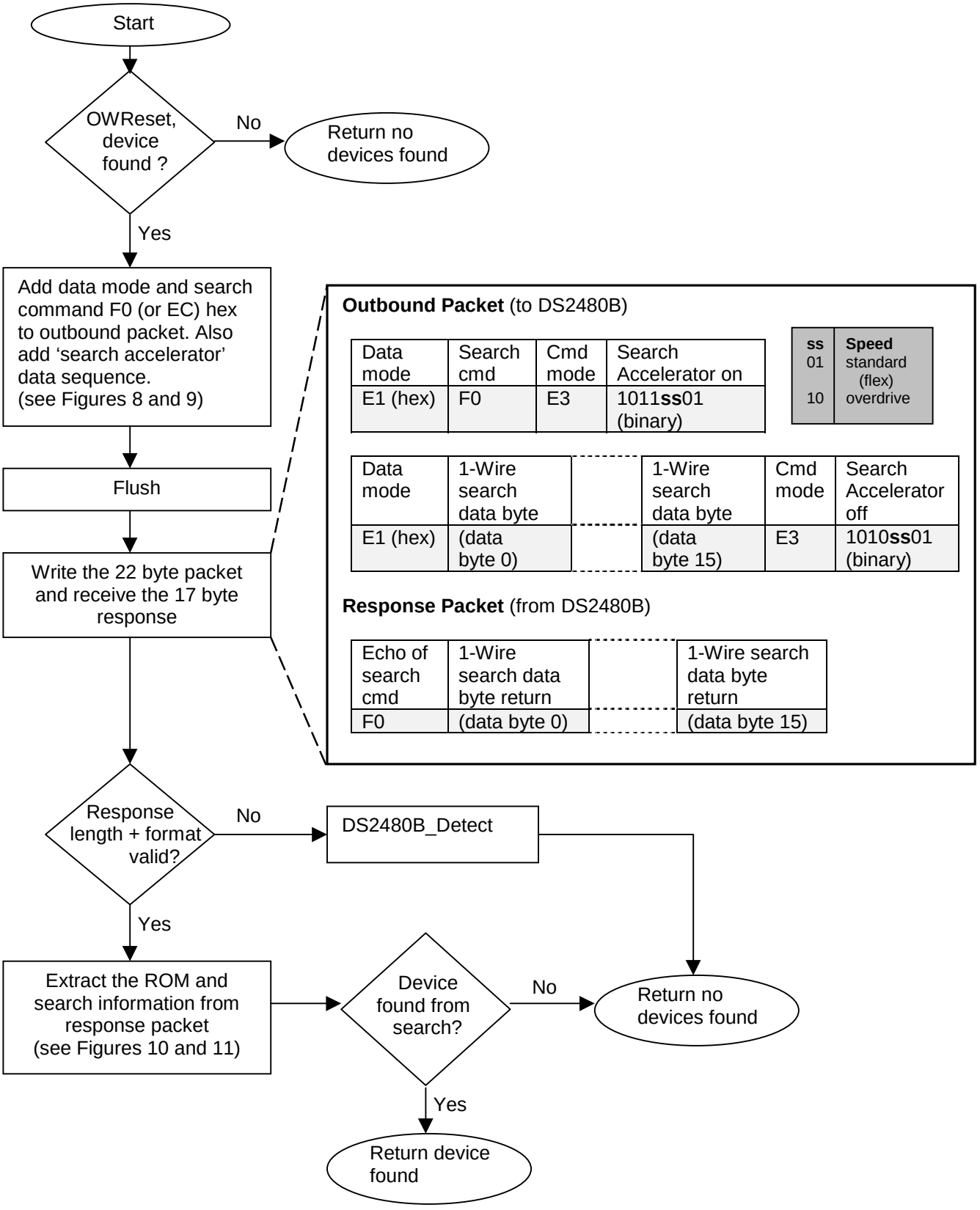
The search algorithm begins with the devices on the 1-Wire being reset using the reset and presence pulse sequence (see *OWReset*). If this is successful then the 1-byte search command is sent (normal F0 hex or alarm EC hex). The search command readies the 1-Wire devices to begin the search.

Following the search command, the actual search begins with all of the participating devices simultaneously sending the first bit (least significant) in their ROM number (also called registration number). As with all 1-Wire communication, the 1-Wire master starts every bit whether it is data to be read or written to the slave devices. When all devices respond at the same time, the result will be a logical AND of the bits sent. After the devices send the first bit of their ROM number, the master initiates the next bit and the devices then send the complement of the first bit. If both bits are zero then there are both 1s and 0s in that bit position. This is called a discrepancy and is a branch point in the search. The 1-Wire master then writes a search direction bit. If the device has that bit value it will continue participating in the search, all other devices go into a wait state. This 'read two bits' and 'write one bit' pattern is then repeated for the remaining 63 bits of the ROM number. See Application Note 187*, 1-Wire Search Algorithm*, at https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/187.html for details on the operation of the 1-Wire search and selective search options.
The basic search operations include finding all devices on the 1-Wire network.
The selective search operations allow searches that find only a particular family of 1-Wire devices.

A large part of the 1-Wire search is performed by the DS2480B. The flow of the search sequence can be seen in Figure 7. The outbound search data is constructed based on the last search (see Figures 8 and 9), the search is performed, and then the search response data is parsed (see Figures 10 and 11).

Care must be taken to not run *OWSearch* in overdrive using this implementation of *OWSpeed* since it uses 115200 baud, which will overflow the DS2480B input buffer. This could be easily modified to reduce the baud rate to 57200 when doing overdrive searches.

OWSEARCH FLOW Figure 7



The state of the 1-Wire search must be maintained between searches to find subsequent devices. The terms representing the search state are presented in Table 5 and coincide with the terms used in Application Note 187, *The 1-Wire Search Algorithm.*

SEARCH STATE Table 5

Term	Description
id_bit_number	The ROM bit number 1 to 64 currently being searched.
LastDeviceFlag	Flag to indicate previous search was the last device.
LastDiscrepancy	Bit index that identifies from which bit the (next) search discrepancy check should start.
LastFamilyDiscrepancy	Bit index that identifies the LastDiscrepancy within the first 8-bit family code of ROM number.
ROM_NO	8-byte buffer that contains the current ROM registration number discovered.
search_direction	Bit value indicating the direction of the search. All devices with this bit stay in the search and the rest go into a wait state for a 1-Wire reset.

1-Wire Search Data Construction

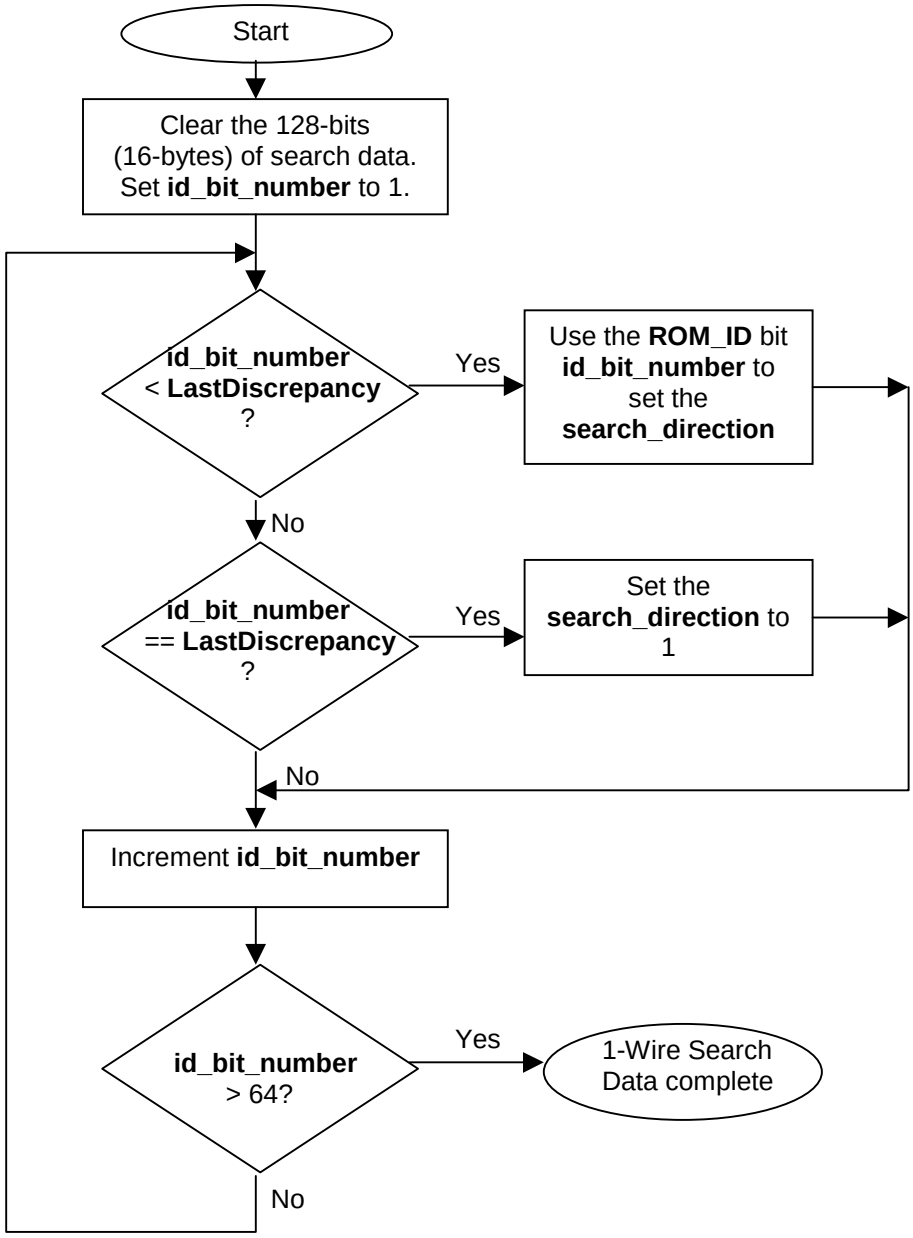
The 16 bytes of 1-Wire search data input can be considered 128 bits of data. The data is grouped into 64 2-bit pairs. The first bit is not used and should be 0. The second bit is the search direction used if a discrepancy is detected. If a discrepancy is not detected then the DS2480B will automatically proceed with the only available path. When constructing the outbound data, set the search direction bits to the ROM_ID bits from the previous search up until the last discrepancy bit position. At that point, set the search direction to one and thereafter set them all to zero. See Figure 8 for the data format and Figure 9 for the construction flow.

OUTBOUND SEARCH DATA Figure 8

128	127	----	**L****	L - 1	----	4	3	2	1
0	0		1	0		**r2**	0	**r1***	0

* **ROM_ID** bits corresponding to the **id_bit_number** from a previous search
** **LastDiscrepancy** bit position

OWSEARCH: OUTBOUND SEARCH DATA CONSTRUCTION Figure 9



1-Wire Search Data Parsing

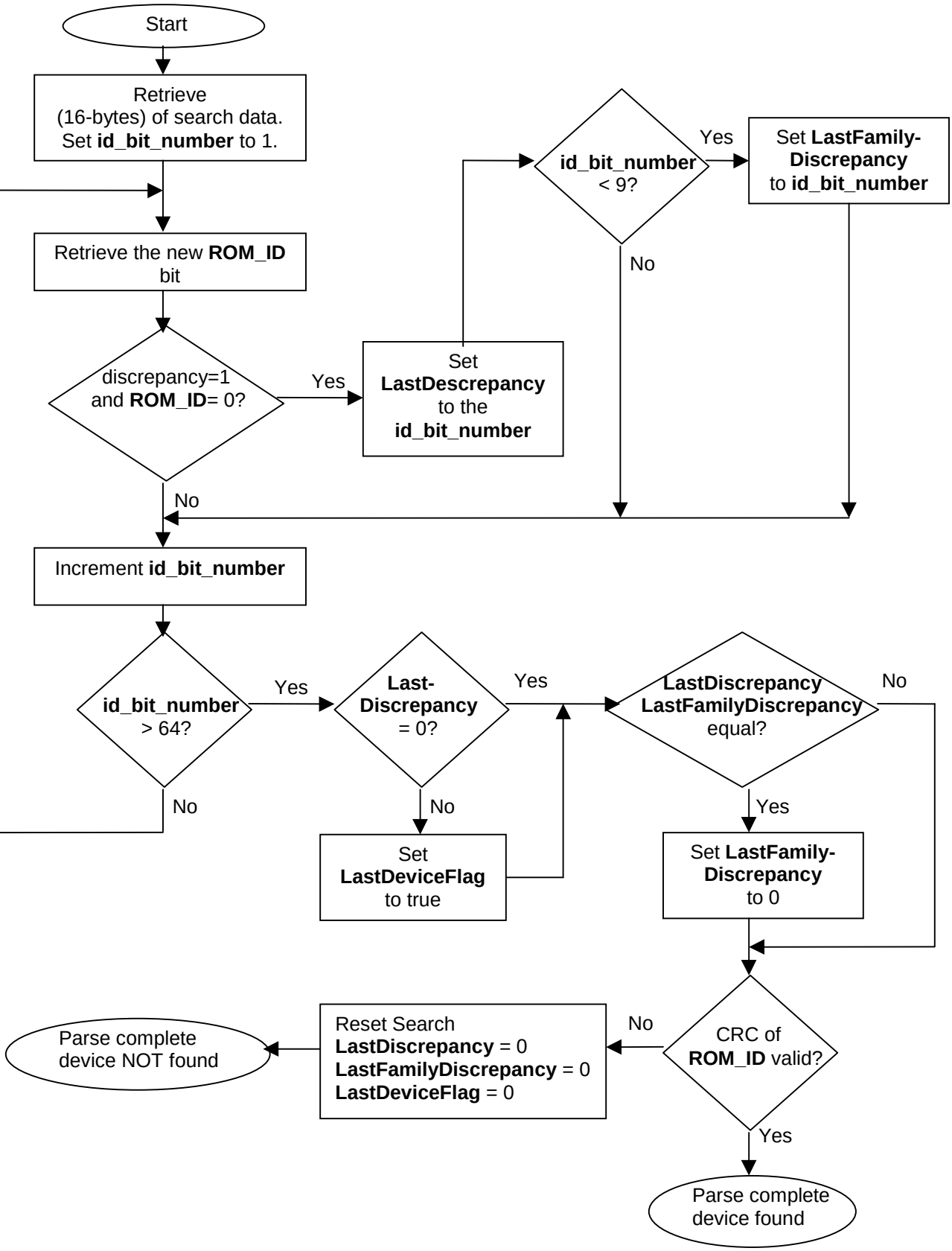
The data resulting from the 1-Wire search from the DS2480B are again 16 bytes of data representing 64 two-bit pairs. The first bit in each pair is a flag indicating if this bit position encounter a discrepancy requiring the use of the search direction bit provided. The second bit in each pair is the search direction taken which is a bit of the resulting ROM number of the device found in the search. The format can be seen in Figure 10. The discrepancy flags and the search direction taken is parsed to set the search state as seen in Figure 11.

SEARCH DATA RESPONSE Figure 10

128	127	----	4	3	2	1
r64	d64		r2	d2	r1*	d1**

* new **ROM_ID** bits corresponding to the **id_bit_number** from this search
** 1 if discrepancy occurred in this bit position, the highest zero position is the new **LastDiscrepancy**

OWSEARCH: RESPONSE SEARCH DATA PARSING Figure 11



OWSpeed

To take advantage of higher throughput of overdrive 1-Wire speed, the baud rate of the serial communication between the host and the DS2480B is increased. This implementation uses only two baud rates: 9600 for standard speed and 115200 for overdrive speed non-search operations. A search accelerator off operation is done to set the DS2480B to the new 1-Wire speed. This is included so that even if the next 1-Wire operation is to communicate bytes in data mode, the correct speed will be used. See the 1-Wire speed flow in Figure 12.

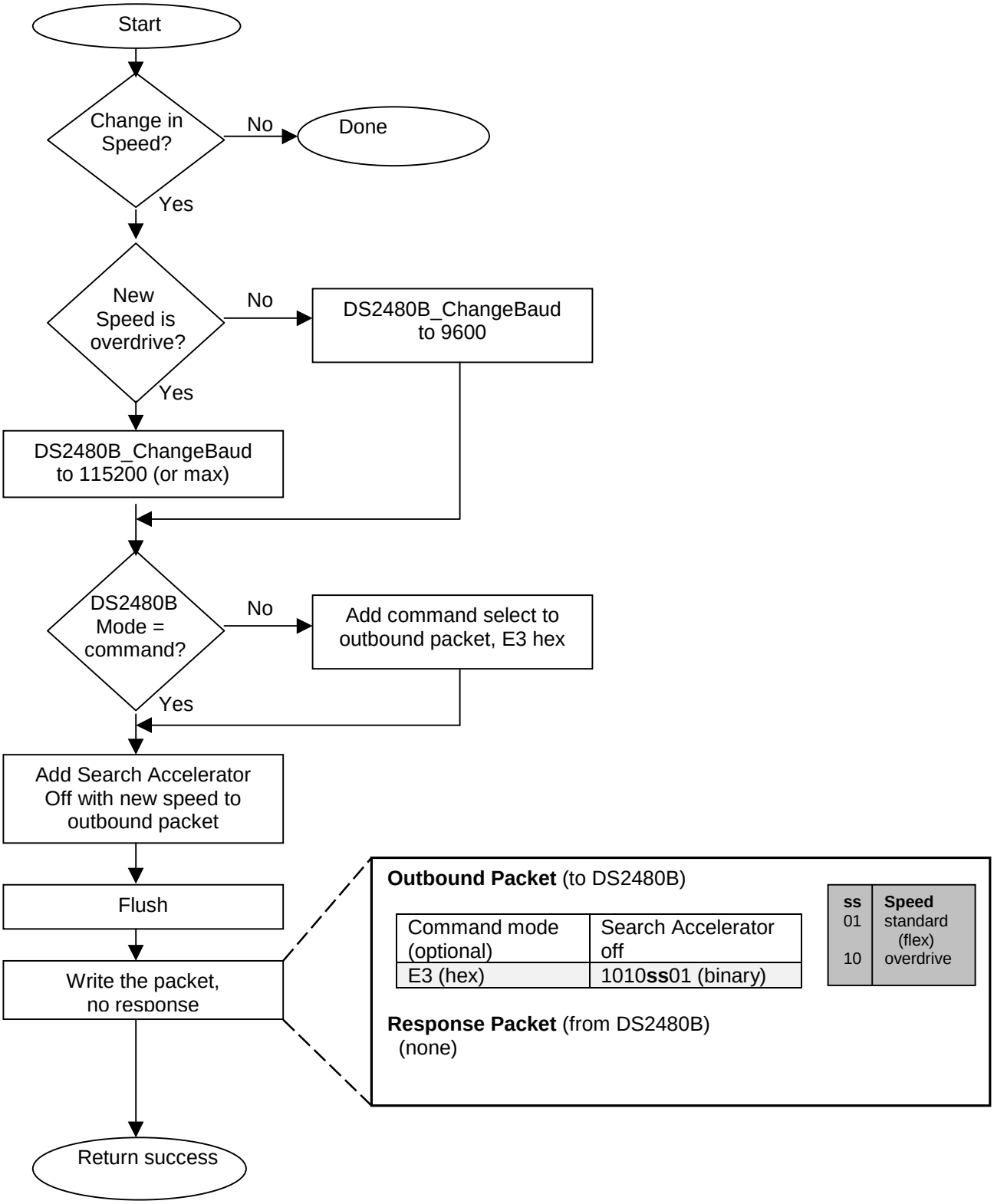
OWLevel

The primary purpose of the *OWLevel* operation is to disable the previously enabled strong pullup power delivery that was initiated from a call to *OWReadBitPower* or *OWWriteBytePower*. The secondary purpose is to manually turn on strong pullup power delivery without the arm feature, although this is not used often. The allowed calls to this operation are *OWLevel(normal)* and *OWLevel(power)*. Note that a call to *OWLevel(normal)* is assumed to be at the beginning of all 1-Wire operations to ensure that the 1-Wire pullup is in the normal state. To disable an armed and current infinite pulse it is necessary to terminate the pulse, start a new pulse without arm enabled and then terminate that pulse. The *OWLevel(power)* operation begins a new non-armed infinite pulse. See Figure 13 for the flow of these two operations.

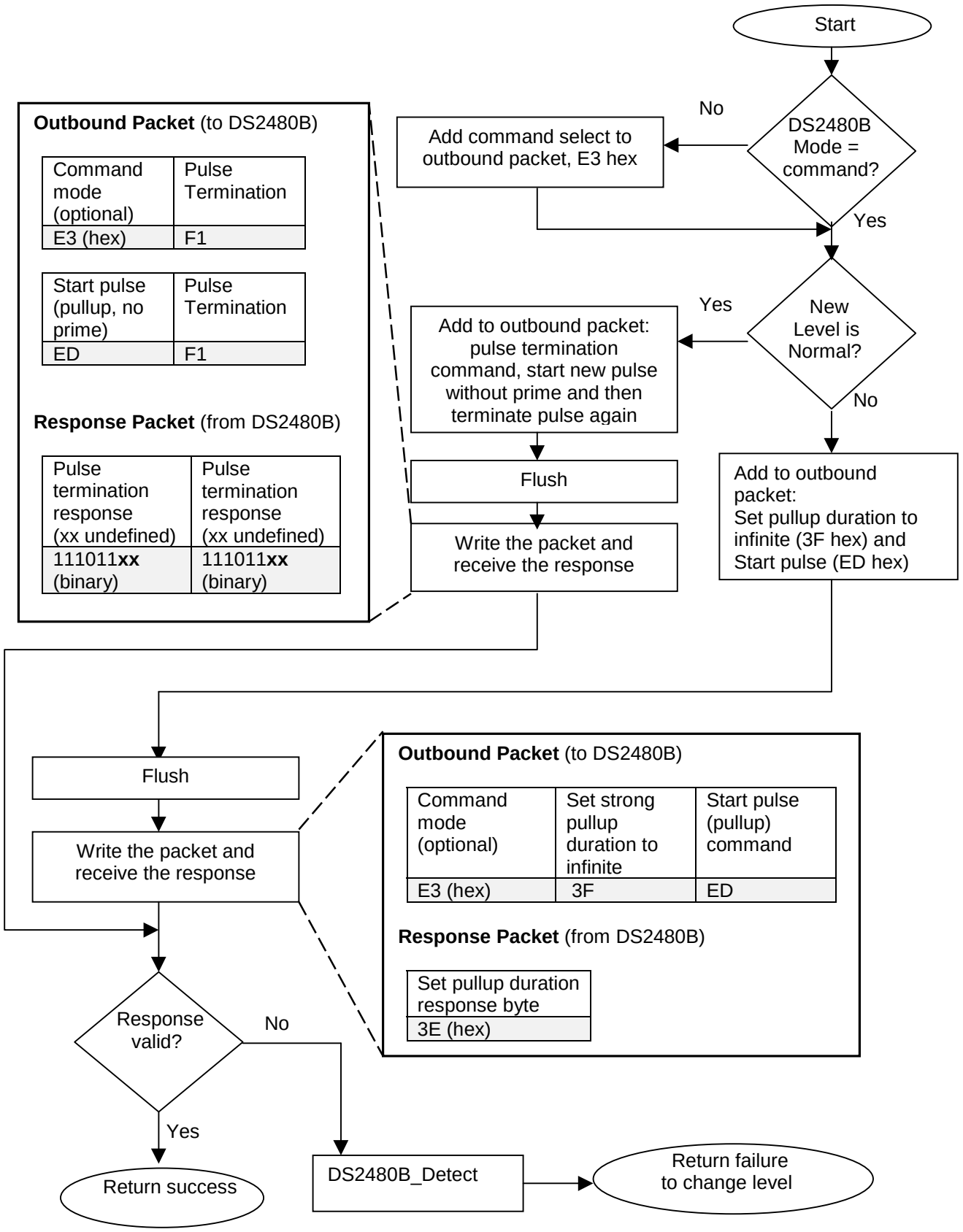
OWProgramPulse

The *OWProgramPulse* operation delivers a 12-Volt programming pulse to the 1-Wire. This operation is used to program 1-Wire EPROM (One-Time-Programmable) memory devices. It checks to see whether the programming voltage is available based on the information derived from the last 1-Wire reset operation *OWReset*. See Figure 14 for the flow of this operation.

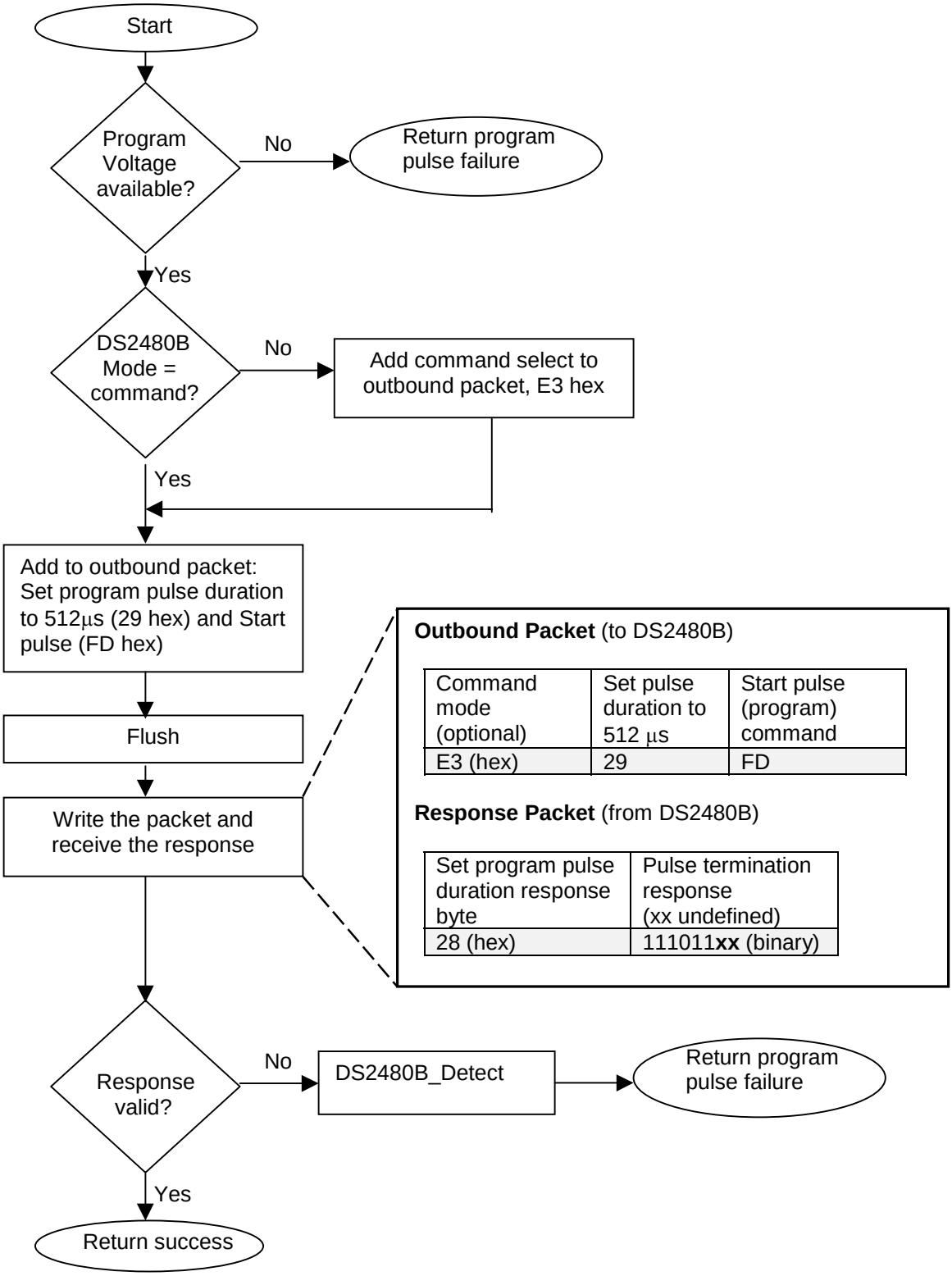
OWSPEED FLOW Figure 12



OWLEVEL FLOW Figure 13



OWPROGRAMPULSE FLOW Figure 14



Outbound Packet (to DS2480B)

Command mode (optional)	Set pulse duration to 512 µs	Start pulse (program) command
E3 (hex)	29	FD

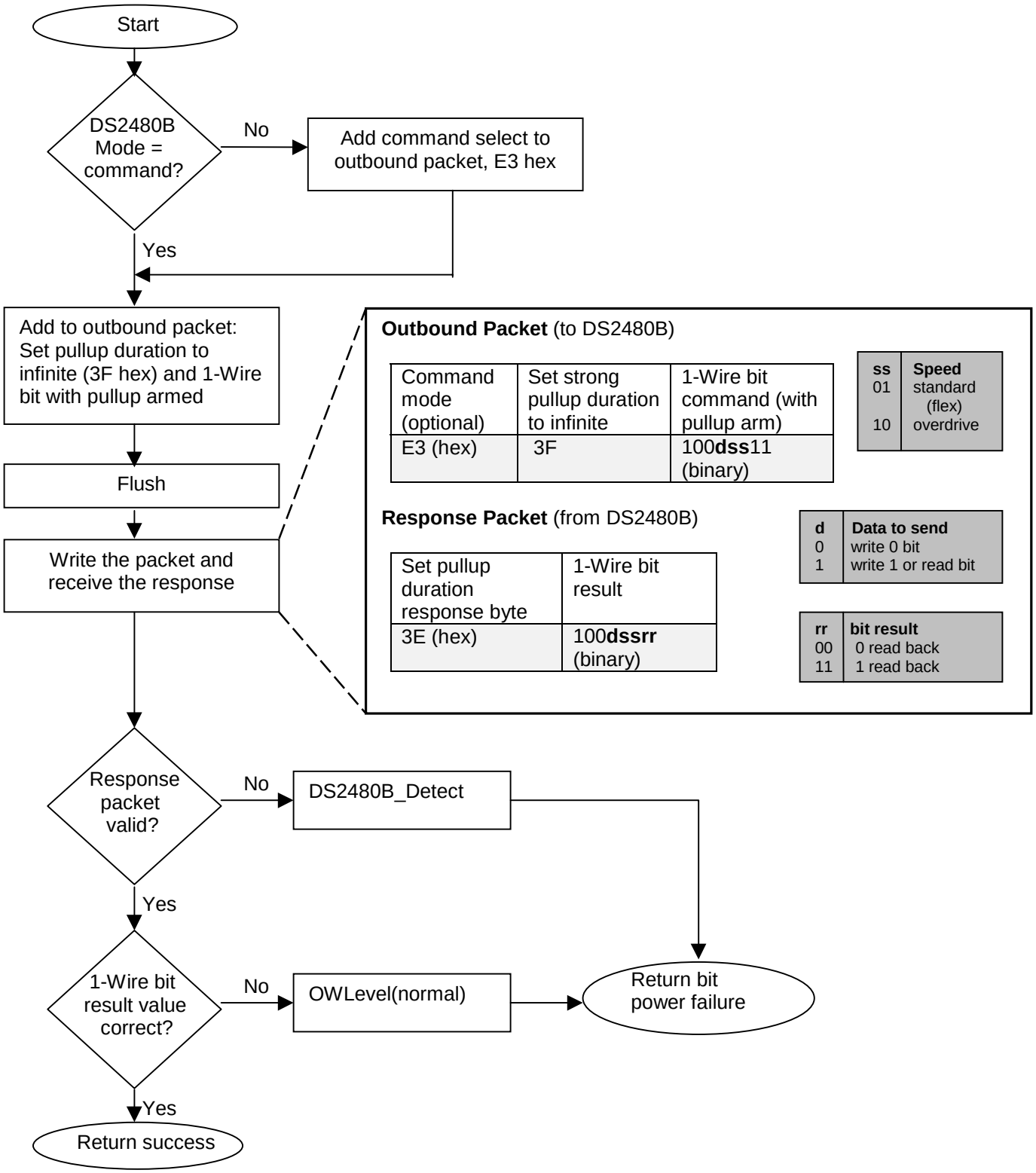
Response Packet (from DS2480B)

Set program pulse duration response byte	Pulse termination response (xx undefined)
28 (hex)	111011**xx** (binary)

OWReadBitPower

The *OWReadBitPower* operation is used exclusively with the Java-powered iButton®. This iButton has a release sequence that must apply power immediately after a confirmation bit. If the confirmation bit is the wrong value then the power is turned back off and the operation fails. If the Java-powered iButton is not going to be used then this operation need not be implemented. To terminate the power delivery after this operation, call *OWLevel(normal)*. See Figure 15 for the flow of this operation.

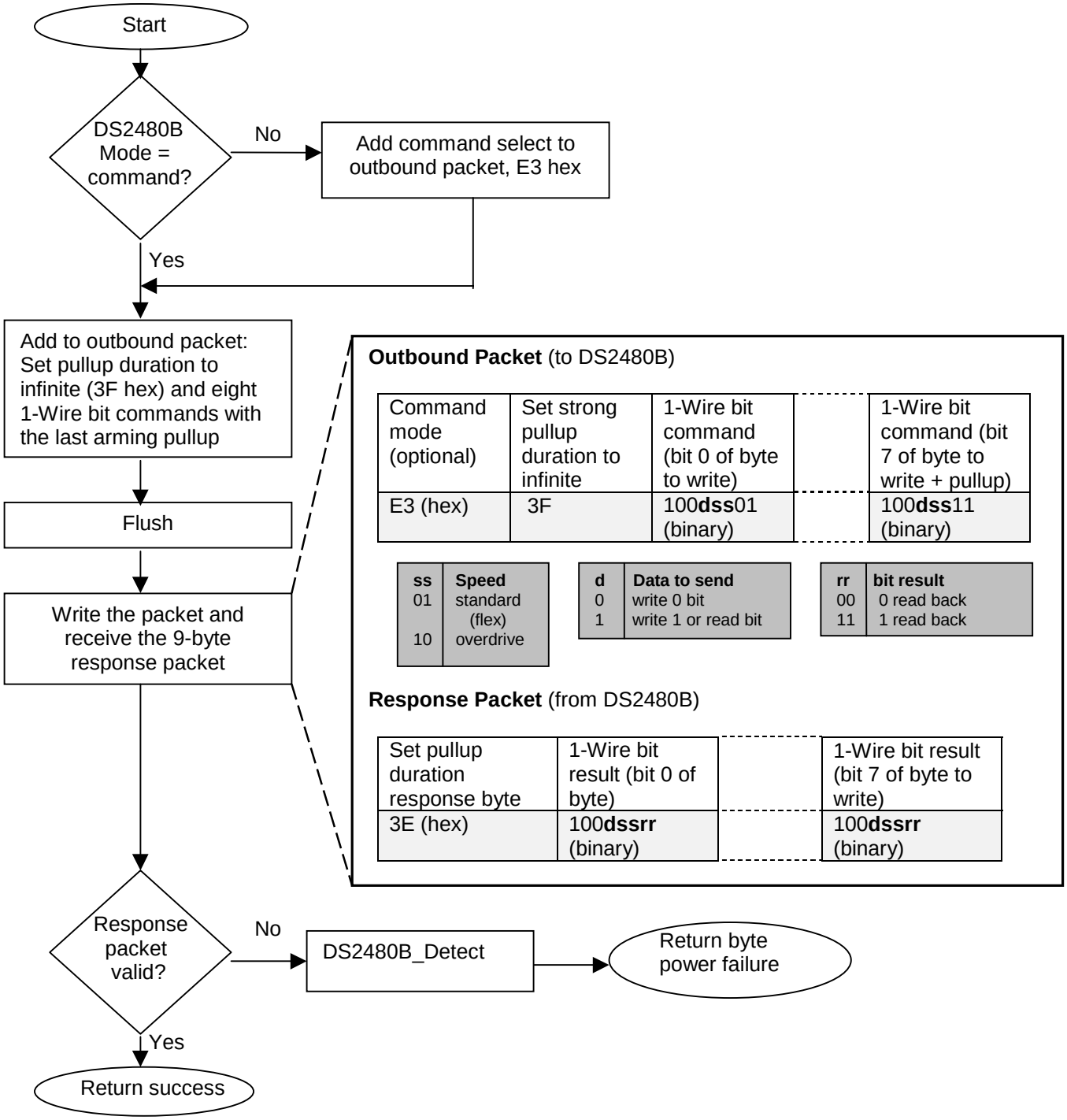
OWREADBITPOWER FLOW Figure 15



OWWriteBytePower

The *OWWriteBytePower* operation applies strong pullup power delivery immediately after a write 1-Wire byte. This is the typical form of power delivery. For example, the DS1920 temperature iButton has a single byte convert temperature command that requires power after the command is complete. To terminate the power delivery after this operation, call *OWLevel(normal)*. See Figure 16 for the flow of this operation. Note that the byte to write is converted to eight single bit operations with the last bit arming the power delivery. This could have been accomplished with a single byte sequence but this was done so that the operation is very similar to the *OWReadBitPower* operation and could potentially be combined.

OWWRITEBYTEPOWER FLOW Figure 16



EXAMPLES

This section has several 1-Wire communication examples using the basic and extended 1-Wire operations.

The overdrive match sequence as seen in Example 2 is used to take an overdrive capable 1-Wire device to overdrive speed. After this sequence has been successfully completed both the DS2480B and the 1-Wire device are operating at overdrive speed and any 1-Wire operation except *OWSearch* can be performed.

OVERDRIVE MATCH PSEUDO CODE Example 2

```
trans_block – temporary transmit buffer, values expressed in hexadecimal notation

// put at 1-Wire speed to normal
OWSpeed(normal)

// reset the 1-Wire bus (at normal speed)
If OWReset = TRUE

   // overdrive match command
   OWWriteByte(69 hex)

   // change 1-Wire speed to overdrive
   OWSpeed(overdrive)

   // send the 1-Wire device ROM number to complete MATCH, ROM is R0...R7
   trans_block = R0,R1,R2,R3,R4,R5,R6,R7
   OWBlock(trans_block)

   // Success
   ...
Else
   // no device present
   ...
EndIf
```

The DS1920 iButton is a temperature reading sensor that performs a temperature conversion when instructed. While a temperature conversion is taking place the 1-Wire master must supply strong pullup power delivery. Example 3 shows the convert temperature sequence using the extended 1-Wire power delivery operations.

DS1920 TEMPERATURE CONVERT PSEUDO CODE Example 3

```
trans_block – temporary transmit buffer, values expressed in hexadecimal notation

// reset the 1-Wire bus If OWReset = TRUE
If OWReset = TRUE

   // sent the MATCH ROM sequence for the device to read, ROM is R0...R7
   trans_block = 55,R0,R1,R2,R3,R4,R5,R6,R7
   OWBlock(trans_block)

   // convert command and apply power
   OWWriteBytePower(44 hex)

   // delay to allow convert to compete
   Delay(1000ms)

   // disable the power delivery
   OWLevel(normal)
```

```
   // verify convert completed
   If OWReadByte == FF hex
      // Success
      ...
   Else
      // convert not complete, fail
      ...
   EndIf

Else
   // no device present
   ...
EndIf
```

One-time-programmable (OTP) EPROM 1-Wire memory devices are written one byte at a time using the sequence as seen in Example 4. The DS2480B must be supplied with a 12V supply for this operation to complete. The availability of the supply is sensed on each called to *OWReset*.

DS1986 EPROM PROGRAMMING PSEUDO CODE Example 4

```
trans_block – temporary transmit buffer, values expressed in hexadecimal notation

// reset the 1-Wire bus
If OWReset = TRUE

   // sent the MATCH ROM sequence for the device to write, ROM is R0...R7
   // with write memory command 0F, and address 0000, and data 66, and read CRC16.
   trans_block = 55,R0,R1,R2,R3,R4,R5,R6,R7,0F,00,00,66,FF,FF
   OWBlock(trans_block)

   // compute CRC16 over last 6 bytes in block to verify data/address set correctly
   If CRC16 correct
      // send the program pulse
      If Not OWProgramPulse
         // Program voltage not available
         ...
      EndIf

      // read back the data for verification
      If OWReadByte != 66
         // Success
         ...
      Else
         // failed to program, page locked, byte already programmed?
         ...
      EndIf
   Else
      // error in transmitting address and data
      ...
   EndIf

Else
   // no device present
   ...
EndIf
```

Example 5 shows a write scratchpad sequence for the DS1996. Note that the entire operation except the reset it lumped together in single block. This is desirable for optimal operation where there is an overhead for each packet sent and received from a serial port.

DS1996 WRITE SCRATCHPAD PSEUDO CODE Example 5

```
trans_block - temporary transmit buffer, values expressed in hexadecimal notation

// reset the 1-Wire bus
If OWReset = TRUE

   // sent the MATCH ROM sequence for the device to write, ROM is R0...R7,
   // the write scratchpad command 0F, target address 0000,
   // and the data (all 66h's for this example)
   trans_block = 55,R0,R1,R2,R3,R4,R5,R6,R7,0F,00,00,66,66,66,66,66,66,66,66,
         66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66,66
   OWBlock(trans_block)

Else
   // no device present
   ...
EndIf
```

CONCLUSION

The DS2480B has successfully been utilized in the following serial to 1-Wire bridge adapters for PCs and workstations: DS9097U-009, DS9097U-S09, DS9097U-E25, DS1411-000, and DS1411-S09. The data sheets for these adapters can be found on Maxim's website at the following link:
http://www.maximintegrated.com

This document has presented a complete 1-Wire interface solution using the DS2480B Serial 1-Wire Line Driver. The provided flow charts are easily implemented on any host system with a minimal serial communications port. A complete 'C' implementation is also available for download from the following link:
http://files.maximintegrated.com/sia_bu/public/an192.zip

Revision History

03/05/02 Version 1.0—Initial release
03/27/03 Version 1.1—Corrections: Search ROM commands corrected to F0 hex. (Figure 4) Changed V_{PP} detection bit in the 1-Wire reset response byte to undefined, ignore response when a DS1994/DS2404 is being read, and correct 1-Wire reset command. Correct Figure 9.