Application Note, V1.0, March 2002

AP3263

TC1765

Understanding the TC1765 BootStrap Loaders

This document describes the functionality of the ASC (asynchronous serial channel), SSC (synchronous serial channel) and CAN (Controller Area Network) bootstrap loader available for TC1765 - Infineon 32-bit Microcontroller.

Microcontrollers

Never stop thinking.

Edition 2002-03

Published by
Infineon Technologies AG
81726 München, Germany

LEGAL DISCLAIMER

THE INFORMATION GIVEN IN THIS APPLICATION NOTE IS GIVEN AS A HINT FOR THE IMPLEMENTATION OF THE INFINEON TECHNOLOGIES COMPONENT ONLY AND SHALL NOT BE REGARDED AS ANY DESCRIPTION OR WARRANTY OF A CERTAIN FUNCTIONALITY, CONDITION OR QUALITY OF THE INFINEON TECHNOLOGIES COMPONENT. THE RECIPIENT OF THIS APPLICATION NOTE MUST VERIFY ANY FUNCTION DESCRIBED HEREIN IN THE REAL APPLICATION. INFINEON TECHNOLOGIES HEREBY DISCLAIMS ANY AND ALL WARRANTIES AND LIABILITIES OF ANY KIND (INCLUDING WITHOUT LIMITATION WARRANTIES OF NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF ANY THIRD PARTY) WITH RESPECT TO ANY AND ALL INFORMATION GIVEN IN THIS APPLICATION NOTE.

Information

For further information on technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements components may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies Office.

Infineon Technologies Components may only be used in life-support devices or systems with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system, or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body, or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

TC1765

Revision History:2002-03V1.0

Previous Version:-

Page	Subjects (major changes since last revision)

We Listen to Your Comments

Any information within this document that you feel is wrong, unclear or missing at all? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:

mcdocu.comments@infineon.com

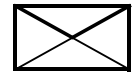


Table of Contents

1 Common BSL Configuration

The section describes the common features of all of the bootstrap loaders.

1.1 Booting Scheme

The bootstrap loader is an integrated mechanism that can be selected via a port configuration during a system start after reset. If the bootstrap loader mode is selected during reset, program execution is started from the Internal Boot ROM.

1.2 Hardware Booting Scheme

The hardware booting scheme uses the state of a number of external pins, sampled and latched with a power-on-reset, to determine the start configuration of the chip. The state of these pins is latched into the Reset Status Register (RST_SR) when the power-on-reset signal (pin $\overline{\text{PORST}}$) is released and transitions to a high level (while $\overline{\text{PORST}}$ is active the latches are transparent). The hardware configuration is determined by the value of bits $\overline{\text{OCDS\_E}}$, $\overline{\text{BRK\_IN}}$ and CFG[2:0]. The latched values can only be changed by another power-on-reset and are used for all hardware-invoked reset options (power-on, hard, watchdog and wake-up reset). **Table 1** shows the Available BootStrap Loaders (BSL) options available in the TC1765.

Table 1 Available Bootstrap Loader ROM Selection

$\overline{\text{OCDSE}}$	$\overline{\text{BRKIN}}$	CFG[2:0]	Type of Boot	Boot Source	PC Start Value
1	1	000_B	ASC Bootstrap Loader	Boot ROM	BFFF FFFC$_H$
1	1	001_B	SSC Bootstrap Loader	Boot ROM	BFFF FFFC$_H$
1	1	010_B	CAN Bootstrap Loader	Boot ROM	BFFF FFFC$_H$

1.3 Entering the Bootstrap Loader Mode

If the selection matches the configuration for the bootstrap loader as shown in **Table 1**. Then program execution begins in the internal 8 KByte boot ROM memory (BFFF FFFF$_H$ - BFFF E000$_H$). The particular BSL routine that runs is determined by the main boot software reading the value of the CFG[2:0] bits and jumping to the respective routine.

1.4 Interrupts

The bootstrap loader software is very simplistic and does not use interrupts. The user should take this into account when attempting to use very high baud rates when connecting to the serial ports.

Note: The user should locate their code assuming a start address of C000 0004$_H$.

1.5 Internal Register Settings

When the TC1765 has entered BSL mode, the following configuration is automatically set.

Table 2 CPU Register Settings[1)]

Register	Setting		Register	Setting
Watchdog Timer	Disabled		BTV	BFFF E200$_H$
ENDINIT Protection	Enabled		ISP	D000 3740$_H$
LCX	000D 000E$_H$		FCX	000D 004C$_H$
BIV	BFFF E000$_H$			

1) All other registers TBD.

Note: The main bootstrap loader routine has configured 64 CSAs.

1.6 TC1765 Operating Frequency

The user must be aware that the TC1765 has no way of determining at what frequency it is operating at. Therefore, the user must have some system knowledge and understanding of the possible communication baud rates can be generated by the TC1765.

The ASC0, SSC0, and TwinCAN modules are all clocked at the same frequency as the $f_{SystemClk}$ and assume the base crystal frequency is 16 MHz.

1.6.1 PLL Factors

The following equation defines the relationship between the f_{OSC} and $f_{SystenClk}$. The PLL “K” factor is controlled via the PLLDIVCLK field in register PLLCLC.

The system clock $f_{SystemClk}$ is:

$$f_{SystemClk} = (10 \bullet f_{OSC})/K$$

An example of a typical clock configuration setting would be as follows:

Table 3 Clock Parameters

PLLDIVCLK	111_B	Unchanged from Reset value, “K” factor is equal to 10
BYPASS	0_B	is at a logic level low, i.e. $f_{SystemClk}$ is supplied from the PLL
f_{OSC}	16 MHz	Base Crystal Frequency

This results in a $f_{SystemClk}$ of 16 MHz.

Note: For more information regarding the clock settings please refer to the TC1765 Users Manual.

2 ASC

2.1 The ASC Bootstrap Loader

The built-in ASC bootstrap loader of the TC1765 provides a mechanism to load the startup program, which is executed after reset, via the asynchronous serial interface. In this case no external (ROM) memory is required for the initialization code. The bootstrap loader moves 128 bytes of code/ data into the internal SPRAM starting at address C000 0004_H.

Most probably the initially loaded routine will load additional code or data. This second receive loop may directly use the pre-initialized ASC interface to receive data and store it to arbitrary user-defined locations.

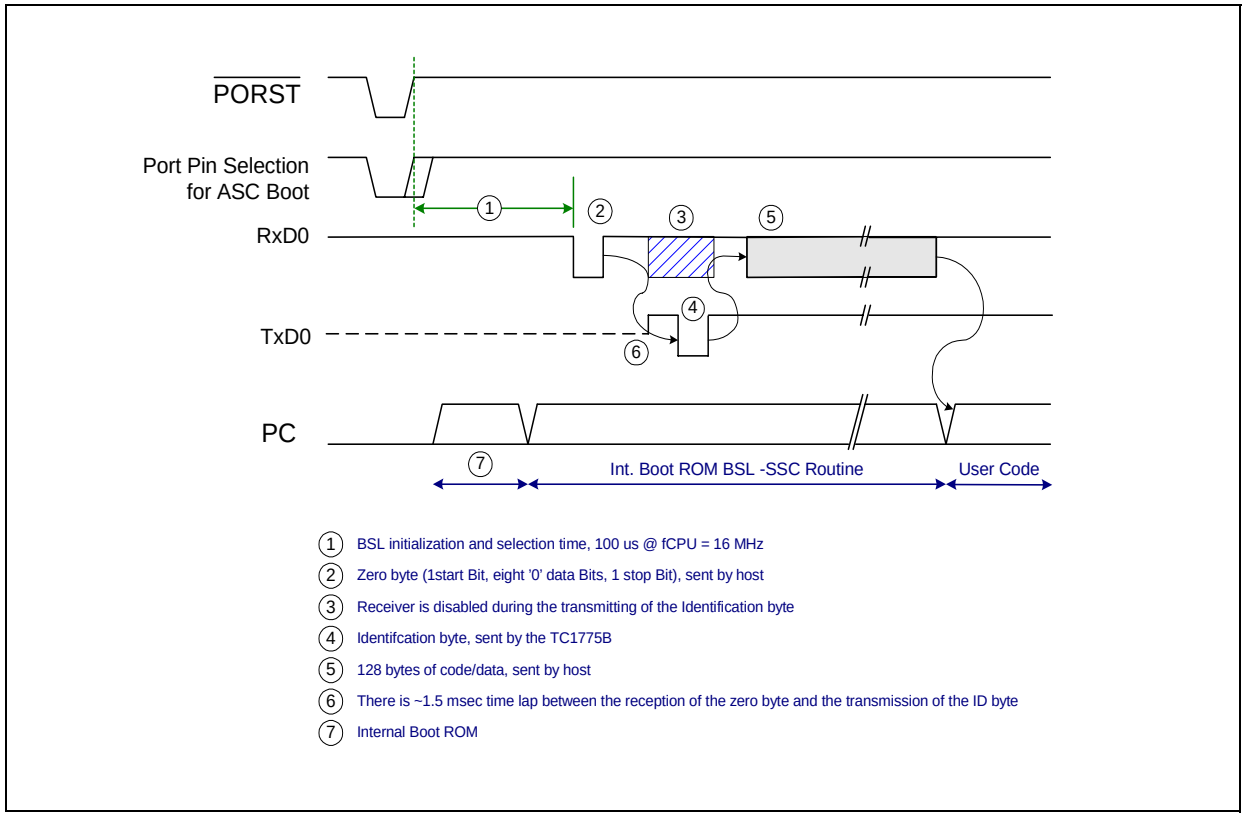


Figure 1 ASC Bootstrap Loader Sequence

2.2 Initialization

The ASC BSL initializes the ASC0 in the following way:

- Receive data pointer is initialized to address C000 0004_H
- baud rate depends on host (auto baud rate detection)
- 8 data bits

- one start/stop bit
- ASC0 interface is selected via port pins
  - P0.7 RxD0
  - P0.8 TxD0

2.3 Entering the ASC Bootstrap Loader

The ASC bootstrap loader code is a routine contained within the boot ROM of the TC1765. TC1765 enters SSC BSL mode when the pin configuration shown in **Table 4**

Table 4 ASC Bootstrap ROM Selection

$\overline{\text{OCDSE}}$	$\overline{\text{BRKIN}}$	CFG [2:0]	Type of Boot	Boot Source	PC Start Value
1	1	000_B	ASC Bootstrap Loader	Boot ROM	BFFF FFFC$_H$

The first task ASC0 needs to perform is to determine the baud rate at which the host is communicating at. This is done by capturing the time period of a zero byte (one start bit and eight “0” data bits) transmitted by the host and received at the ASC0 receive pin. Therefore, a software polling loop begins by first waiting for the ASC0 receive pin to remain at a high level. Once a high level has been detected the software now begins looking for a falling edge. After a falling edge has been detected the system timer is read and the software now looks for a rising edge.

When the rising edge is found the system timer is read again and the difference between the two readings is calculated. This value represents the time period for one zero byte transmitted by the host and is the basis for calculating the host baud rate with respect to the f_{SYSCLK} clock.

The next step is to enable the ASC clock generator. This requires a password unlock sequence to be performed to the WDT to gain write access to the ASC CLC Control register ($f_{SYSCLK} = f_{ASC}$). After writing the ASC CLC register the WDT is then locked again.

The ASC boot loader continues by initializing the serial port pins and calculating the baud rate. The baud rate is calculated by searching for the best values for the Fraction Divide Value (FDV) and Baud rate Generator (BG) registers.

Figure 2, represents the block diagram of the ASC Baud Rate Generator.

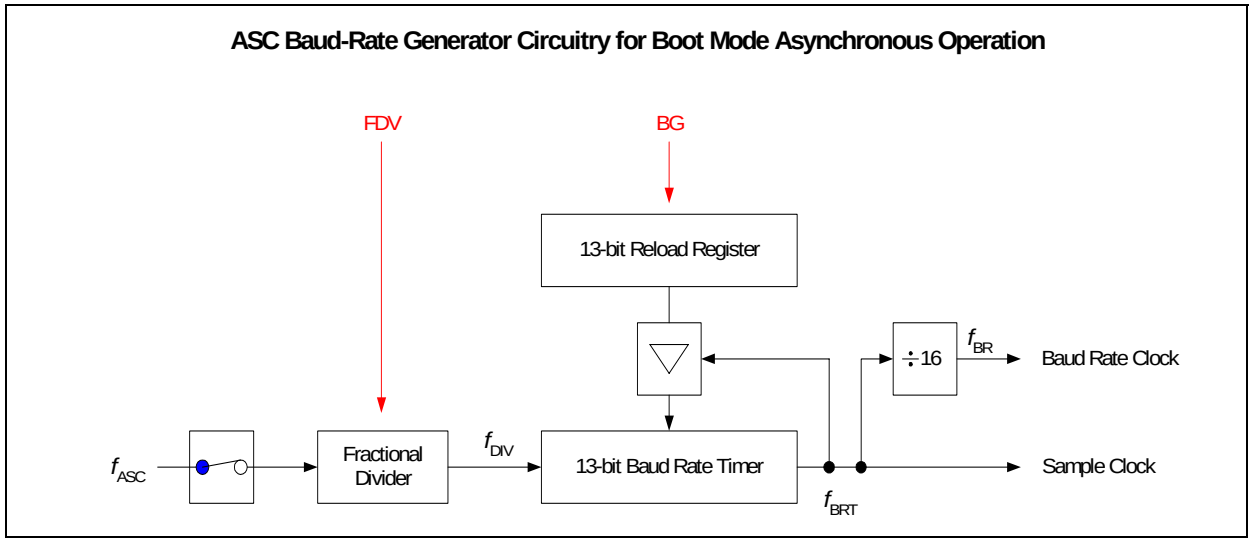


Figure 2 ASC Baud Rate Generator

The software must now determine the values to be written in the Fractional Divide Value register and the Baud rate Generator register. The Baud rate is found by using min/max boundaries to find the best fit value for BG as shown in the following formulas.

Timer Value = Measured system timer tick for a zero byte transmission

$$baud\_rate_{ASC} = \frac{9 \cdot f_{ASC}}{TimerValue}$$

Therefore substituting for *baud_bate*$_{ASC}$ in the equations and solving for *BG*:

$$\frac{9}{TimerValue} = \frac{FDV}{(512 \cdot 16 \cdot (BG+1))}$$

$$\frac{9 \cdot 512 \cdot 16}{TimerValue} = \frac{FDV}{BG+1}$$

$$FDV = \frac{73728 \cdot BG}{TimerValue} + \frac{73728}{TimerValue}$$

$$\frac{73728 \cdot BG}{TimerValue} = FDV - \frac{73728}{TimerValue}$$

$$73728 \cdot BG = FDV \cdot TimerValue - 73728$$

$$BG = \frac{FDV \cdot TimerValue}{73728} - 1$$

Next the ASC0 is initialized (receive pin remains disabled) to the baud rate of the host, an identification byte ($D5_H$) is returned to the host indicating the device is ready to accept a data transfer from the host of **exactly 128 bytes**. The ASC0 receive pin is enabled after the identification byte has been transmitted.

The software now enters two receive loops. The inner loop is waiting until it has received four bytes. The outer loop will write (word) the word access to the program memory. This process repeats until 128 bytes have been received.

Note: The ASC bootstrap loader expects a block size of exactly 128 bytes (32 words).

2.4 Exiting the ASC Bootstrap Loader

Once 128 bytes have been received and written to program memory a “Jump Indirect” command (JI) is performed to the memory location C000 0004_H and program execution begins at this point (user program).

2.5 ASC Boot Algorithm for the Internal ROM Program Flow

Figure 3 shows a flowchart of the ASC boot algorithm.

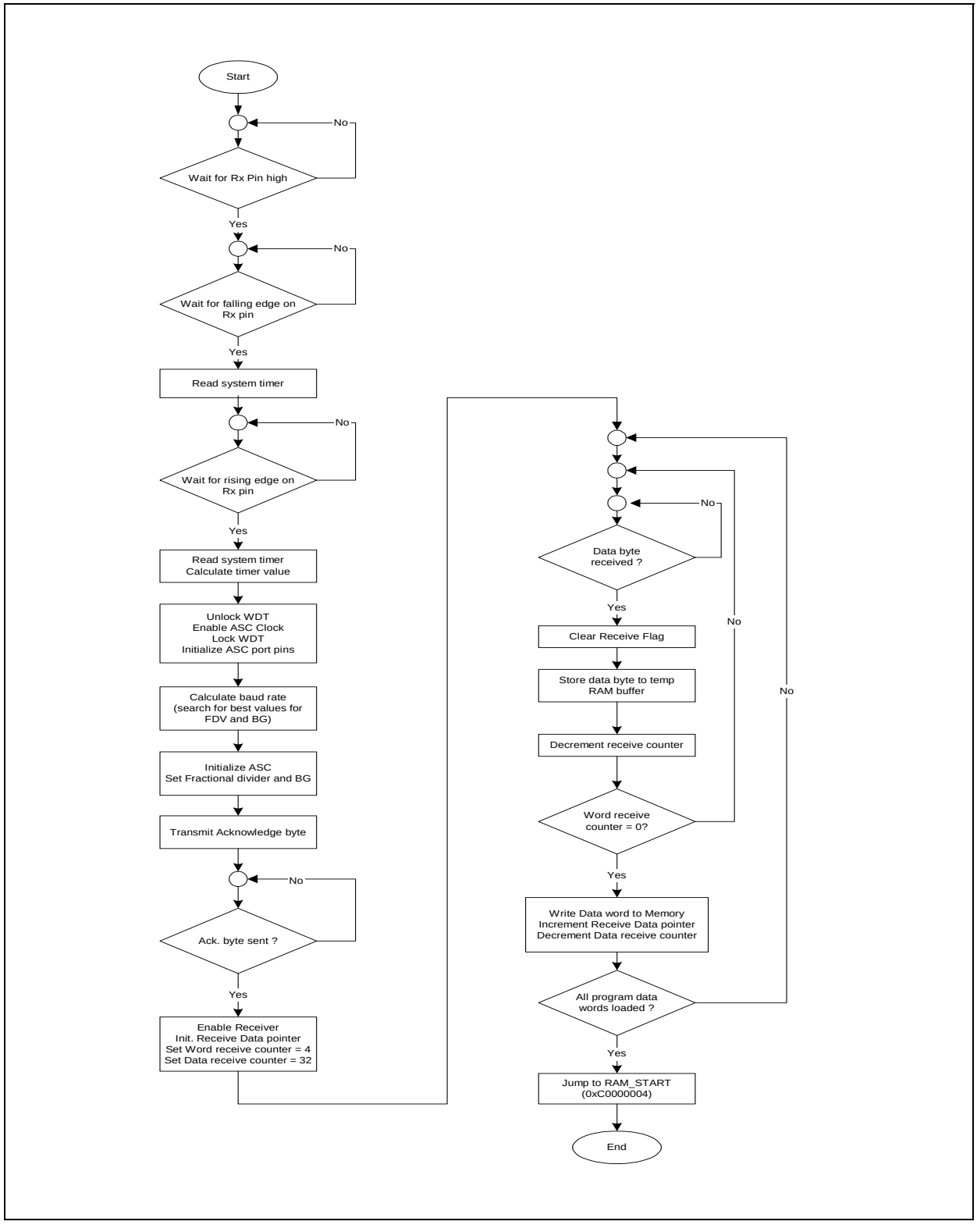


Figure 3 ASC Boot Algorithm

2.6 ASC Timing

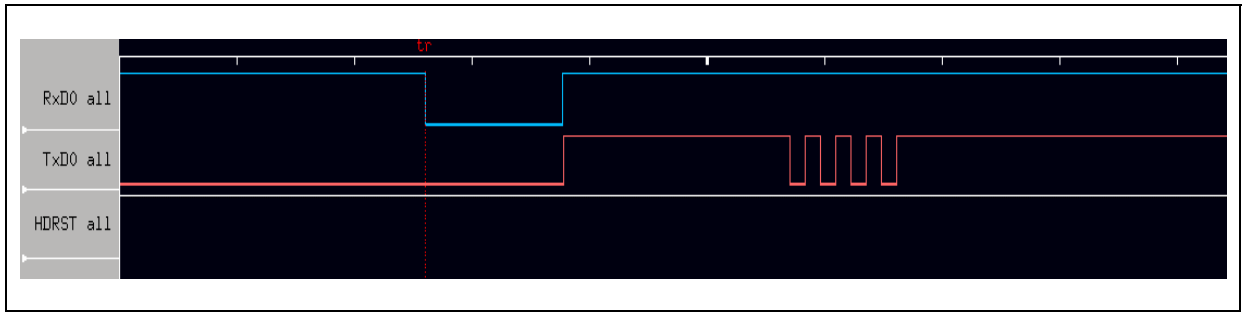


Figure 4 Logic Analyzer Trace of the actual data transfer

Typical Timing (Bus Clock 16 KHz, 9600 Baud)

Boot Initialization (Host must wait before sending the zero byte from the rising edge of PORST)

~ 100 µsec

From rising edge of RxD (host last data bit) to the falling edge of TxD (Start bit from the TC1765 sending the ID byte).

~ 1.5 msec

Delay from the reception of the ID byte to the host downloading data

~ 100 µsec

Between byte transfers: at high baud rate the user may need to add inter-byte delays

3 SSC

3.1 The Synchronous Serial Channel (SSC) Bootstrap Loader

The built-in Synchronous Serial Channel bootstrap loader of the TC1765 provides a mechanism to load a user defined program via the SSC0 interface. The bootstrap loader moves code/data into the internal Scratch Pad RAM (SPRAM) starting at location “C000 0004_H”.

The default SSC BSL configuration is assumed to have an SPI compatible EEPROM (25xxx series) connected to the TC1765’s SSC0 port. The SSC BSL supports memory devices with 8-bit or 16-bit addressing and any other possibility can be assumed by the user if they understand this default configuration.

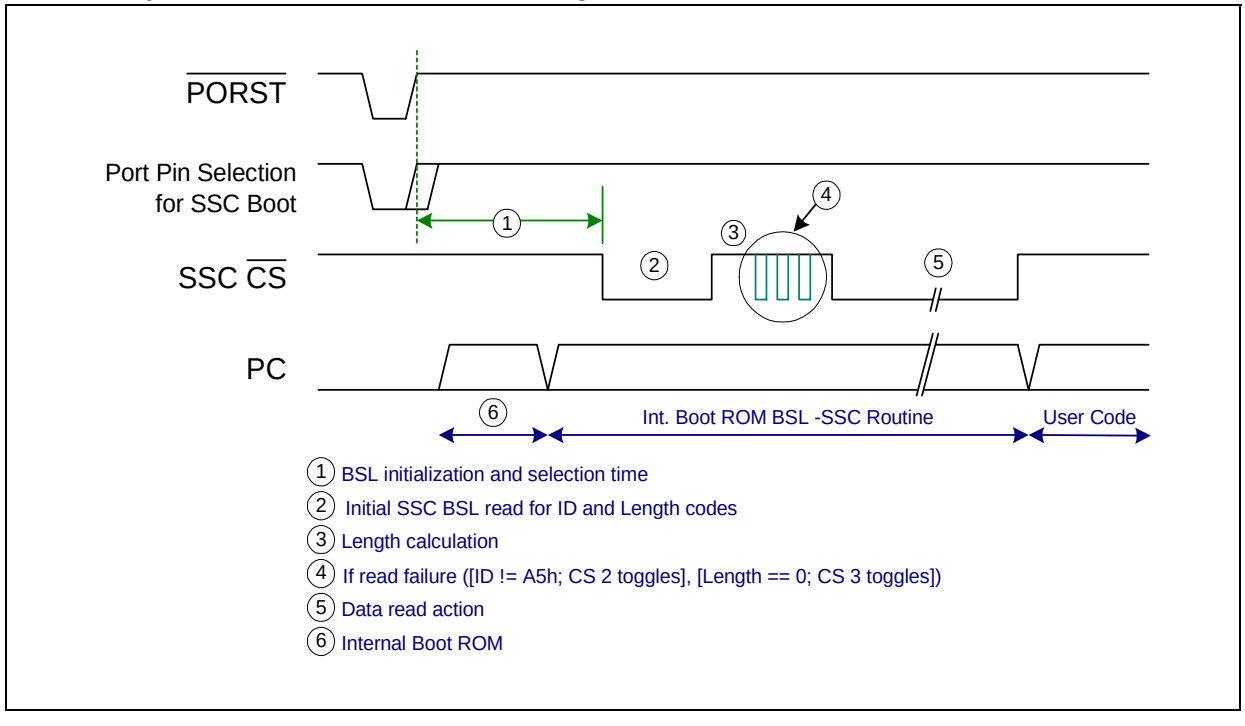


Figure 5 SSC Bootstrap Loader Sequence

3.2 Initialization

The SSC BSL initializes the SSC0 in the following way:

- CPU is master for master mode
- SPI mode 0 (0,0)
- 1 MHz clock rate (16 MHz external clock)
- Data length 8 bits
- Port pin assignment (4 pins)
- Receive data pointer set to address C000 0004_H
- SPI memory chip-select via port pin
  - P0.9 SCLK
  - P0.10 MRST
  - P0.11 MTSR
  - P4.0 $\overline{CS}$

3.3 Entering the SSC Bootstrap Loader

The SSC bootstrap loader code is stored in and is part of the TC1765 boot ROM. The TC1765 enters SSC BSL mode when the pin configuration shown in **Table 5** following a PORST.

Table 5 SSC Bootstrap ROM Selection

$\overline{OCDSE}$	$\overline{BRKIN}$	CFG [2:0]	Type of Boot	Boot Source	PC Start Value
1	1	001_B	SSC Bootstrap Loader	Boot ROM	BFFF $FFFC_H$

3.4 Loading the Startup Code

The port pins for the SSC0 and $\overline{CS}$ are setup along with the SSC operating parameters. The boot software attempts to set the SSC clock frequency to a nominal rate of 1 MHz (assuming a base crystal frequency of 16 MHz).

The SSC BSL will basically perform two read accesses (see **Figure 7** and **Figure 8**). The first read access is to determine the addressing mode and the number of bytes to read. The second access is to actually load the user program into its internal program RAM. There is also the possibility of detecting two data format errors. The errors are either for an incorrect Identification byte or zero data length. The user is notified of an error on the $\overline{CS}$ line. The toggling of the $\overline{CS}$ line between the two read access would indicate that an error occurred during the initial read access.

Both read access begin by the SSC sending a read command (data value = 3) with a memory start address of zero. The SSC BSL determines the type of memory (8- or /16-bit addressing) by checking at which position the memory identifier byte is located. An example of the memory map of an EEPROM device is shown in **Table 6**. For memory types with 8-bit addressing this will happen after the first address byte, for memory types with 16-bit addressing this happens after the second address byte is read. If the correct

memory identifier byte is not equal to $5A_H$ then the CS line is quickly toggled twice and then the software enters an endless loop. Assuming there was not an error the data byte following the memory identifier is assumed to be the size index byte. The size index informs the bootstrap loader of the number of bytes multiplies by 16 that are to be received. This allows block of up to 2040 (32-bit) or 4080 (16-bit) instructions can be loaded into memory. This process continues until all bytes have been read from the serial memory.

Table 6 Content of the EEPROM

EEPROM Address	Data value	Meaning
0x0000	0x5A	Memory Identifier (ID)
0x0001	0x01...0xFF	Size Index (Number of bytes to load * 16)
0x0002...0x1FE2	0x00...0xFF	User Code/Data

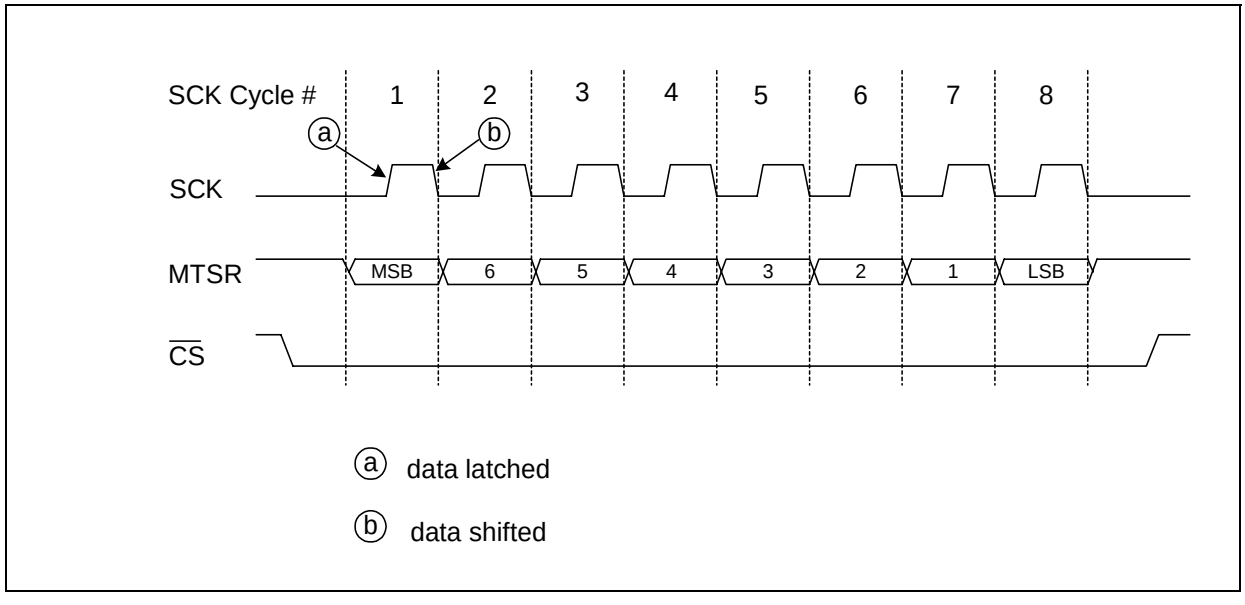


Figure 6 Example of SPI Mode 0

Note: The SSC BSL uses mode 0 to communicate thereby bit PH is set to “1” and bit PO is set to ”0” in register SSCx_CON.

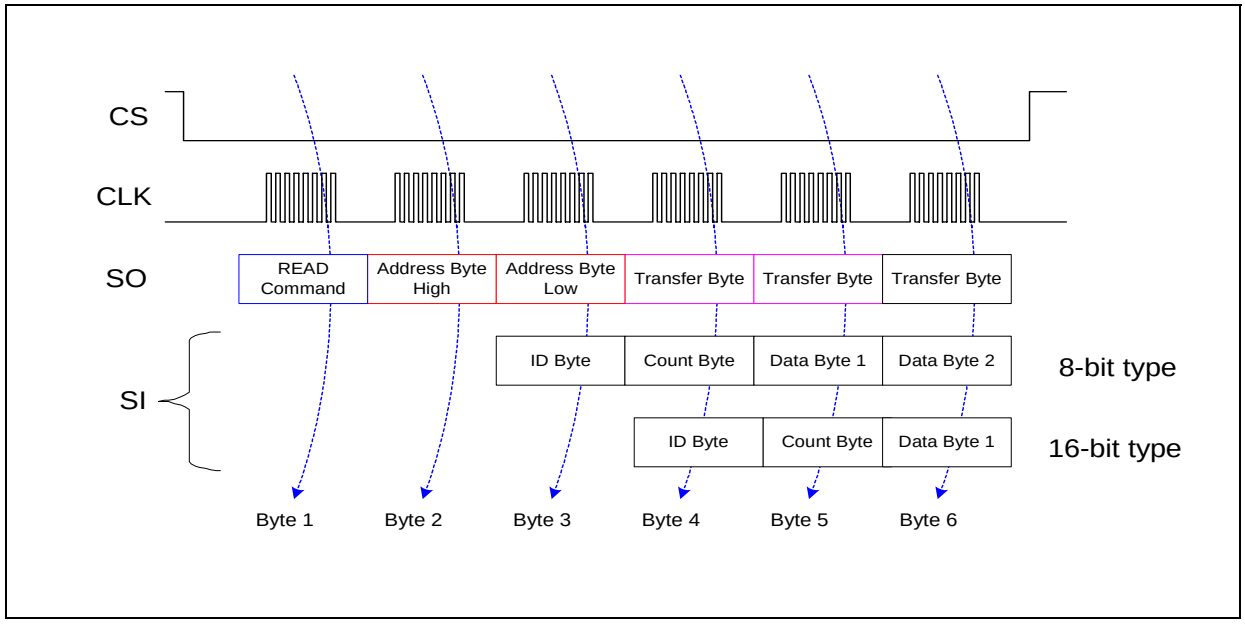


Figure 7 Initial SSC BSL data transfer

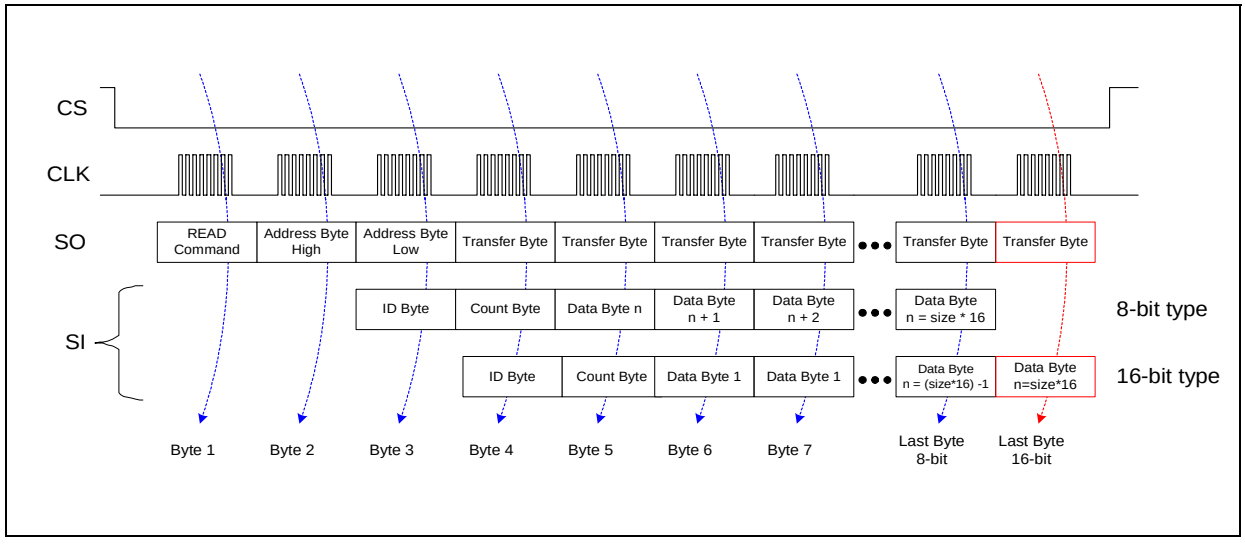


Figure 8 Second SSC BSL data transfer

3.5 Baud rate generation for the BSL

The BSL has no way to determine its base clock frequency and therefore assumes that this is 16 MHz. Additionally, the K factor of the PLL is not modified by the BSL so it remains at its default value of seven. The BSL will automatically enable the SSC clock and the set the value of the SSC Clock divider to 1 (unity) so that the SSC clock frequency is equal to the system clock frequency f_{SYSCLK}. The system clock frequency for TC1765 is determined from the following formula:

$$f_{SYSCLK} = \frac{10}{K} \bullet f_{OSC}$$

The calculation of the reload value is done by:

$$
 = \left(\frac{f_{SSC}}{2 \cdot baud\_rate_{SSC}} \right) - 1$$

The actual $baud\_rate_{SSC}$ value must be determined from the following formula:

$$baud\_rate_{SSC} = \frac{f_{SSC}}{2 \cdot (
 + 1)}$$

3.6 Exiting the SSC Bootstrap Loader

Once all bytes (size index * 16) have been read and written to program memory a “Jump Indirect” command (JI) is performed to the memory location C000 0004_H and program execution begins at this point (user program).

Note: The user should locate their program to a start address of C000 0004_H.

3.7 SSC Boot Algorithm

Figure 9 and **Figure 10** show a flowchart of the SSC boot algorithm.

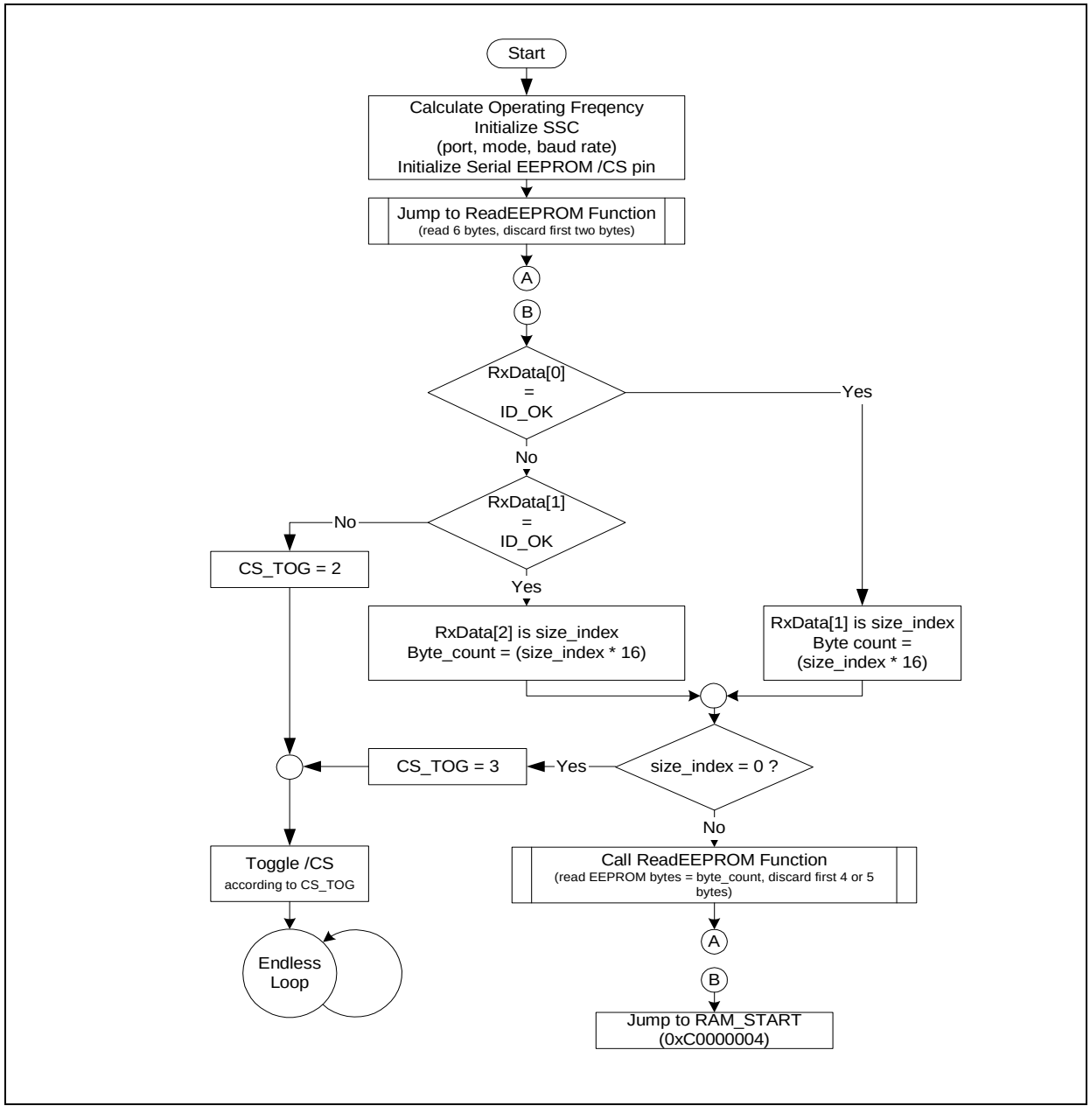


Figure 9 SSC Boot Algorithm (Sheet 1/2)

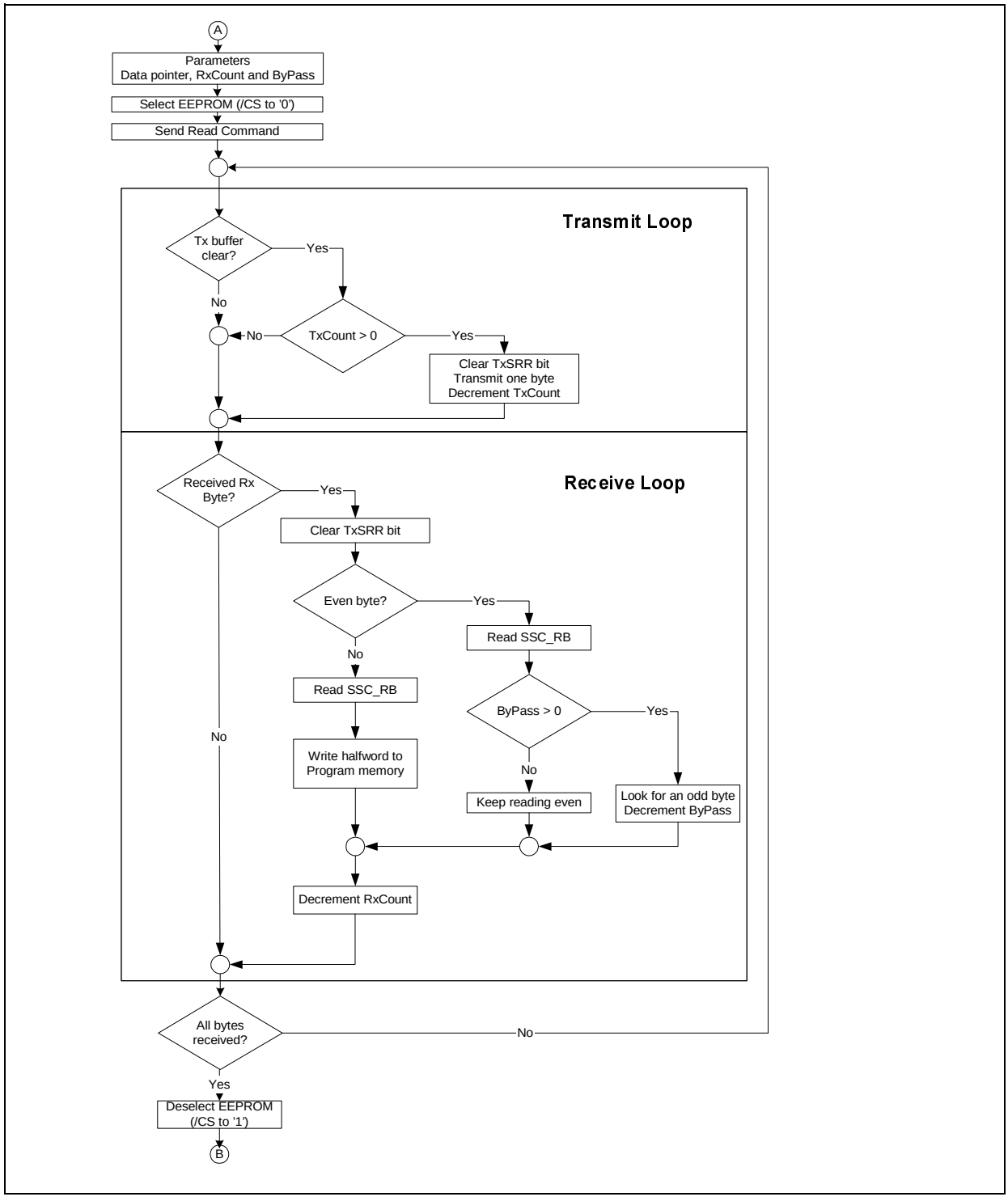


Figure 10 SSC Boot Algorithm (Sheet 2/2)

3.8 SSC BSL Data Transfer Timing

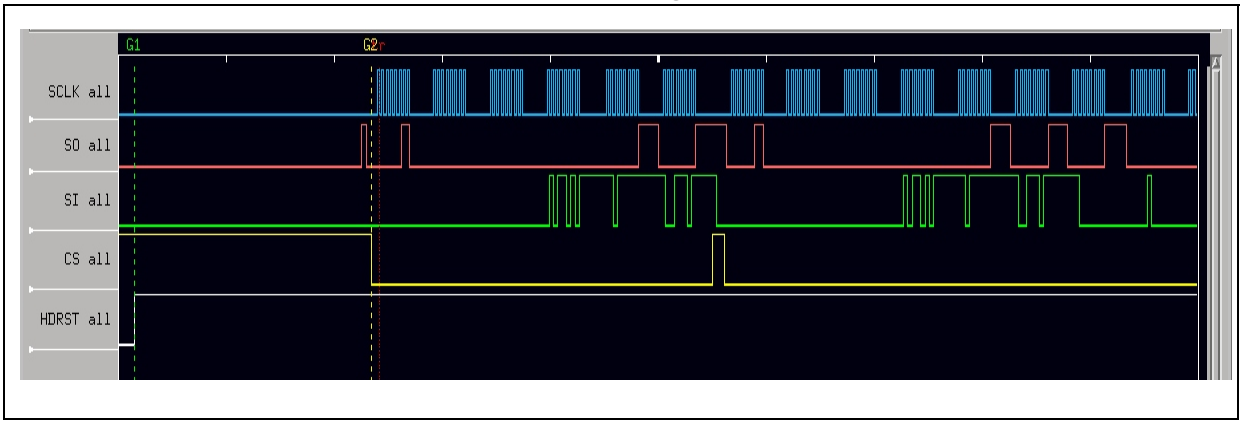


Figure 11 Logic Trace of the start of the SSC BSL data sequence.

Typical SSC BSL Timing (Bus Clock 16 kHz)

Boot Initialization (rising edge of $\overline{\text{HDRST}}$ and the falling edge of $\overline{\text{CS}}$)

~ 60 µsec

From rising edge CS to the first rising edge of SCLK.

~ 15 µsec

Delay between byte transfers

~ 12 µsec

Time from the falling edge of the last SCLK and the rising edge of CS.

~ 4 µsec

CS high time between the first and second read assesses.

~ 3 µsec

SPI Mode Definitions:

Clock Polarity is defined by symbols **CPOL** and **PO**.

Clock Phase is defined by symbols **CPHA** and **PH**. Please note that Infineon defines the value of the clock phase to be inverted to the normally accepted industry standard (see **Figure 12**).

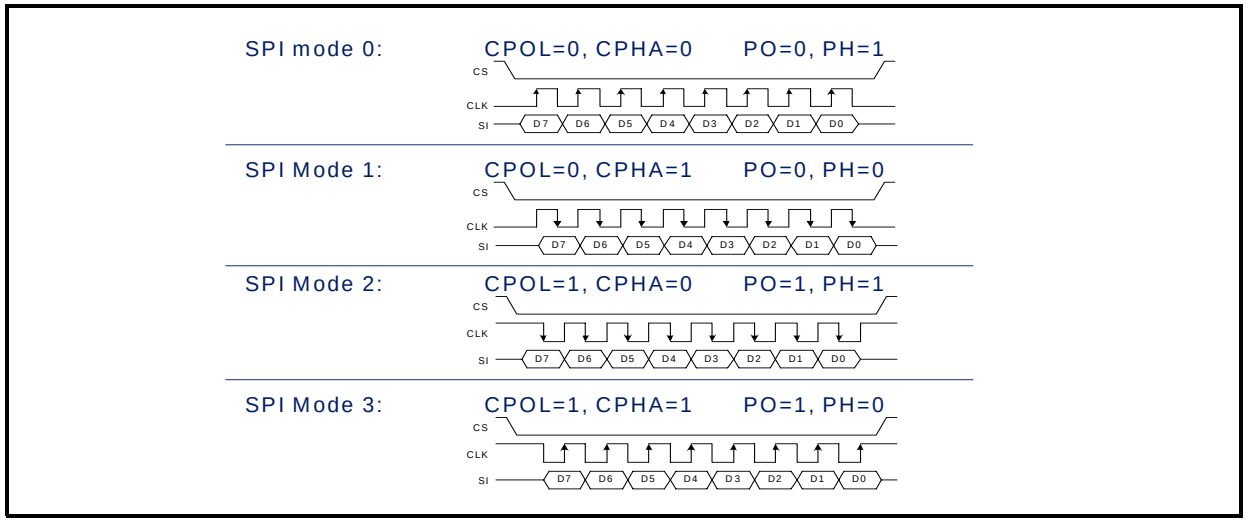


Figure 12 Industry accepted SPI transfer modes and settings compared to Infineon's definition

4 CAN

4.1 The CAN Bootstrap Loader

The built-in CAN bootstrap loader of the TC1765 provides a mechanism to load a program via the TwinCAN module (port pins P0[12:13]). The bootstrap loader is an integrated mechanism that can be selected via a port configuration during a system start after reset. If the bootstrap loader mode is selected during reset, program execution is started from the Internal Boot ROM. The bootstrap loader program will configure its CAN module to the baud rate of the Host in order for communications to transpire. Once communication has been established, the bootstrap loader receives a Host defined variable number of messages for downloading of the code/data. The received code/data is sequentially written to the on-board Local Code Scratch Pad RAM (SPRAM). In the TC1765, there are 16 Kbytes of SPRAM available for program execution. After the download has been completed, the program execution continues by jumping to the start of the SPRAM (C000 0004_H).

The bootstrap loader mechanism may be used for standard system startup as well as for special occasions like system maintenance (firmware update), end-of-line programming or testing.

4.2 TC1765 Operating Frequency

The user must be aware that the TC1765 has no way of determining at what frequency it is operating at. Therefore, the user must have some system knowledge and understanding of what CAN baud rates can be detected from the Host.

A PC based software executable is provided which determines what baud rates the TC1765 is capable of receiving based on its system clock frequency ($f_{SystemClk}$).

4.3 Initialization

The CAN BSL initializes the TwinCAN in the following way:

- The CAN module 0 is used but not initialized until a message is detected
- Message Object 0 is configured to receive a standard CAN frame with a message identifier of 555_H.
- Receive data pointer set to address C000 0004_H
- CAN port pins
  - –P0.12 RxCAN0
  - –P0.13 TxCAN0

Note: It is recommend that Port0 have no floating inputs or any of its pins changing states other than P0.12. The CAN BSL software determines the bit timing by reading the entire port and using it for sampling time calculations.

4.4 Measurement of dominant CAN bits

The System Timer is used for the measurement of a dominant bit on the CAN bus. The System Timer is running at the same frequency as the CPU and thus supports the highest possible resolution.

During this time the CAN module is disabled and the TC1765 software is polling the RxCAN0 (P0.12) pin to detect pulse bus periods (capturing the time between any edge). This process will continue until 100 pulses have been measured. Then, a software algorithm will analyze the data to determine the smallest pulse. From this, the data is further analyzed to obtain an average of the smallest pulses. This time period is assumed to be one CAN bit time and, therefore, it is used for the initial setting for detecting the correct baud rate.

4.5 Requirements of the Host

The host will initiate communication with the TC1765 by transmitting a standard CAN message (11-bit Identifier). However, for the TC1765 to properly determine the message baud rate, certain restrictions are imposed on the Host.

- There is a Point-to-Point connection between the Host and TC1765
- The sample point of a CAN bit is at 80%
- A baud rate is used which the TC1765 is capable of detecting
- The first message the Host sends is of the format below
  - –The identifier for the standard message is 555_H
  - –The data length code is set to 6
  - –Data bytes 0 and 1 are the values for the TC1765's Bit Timing Register (BTR)
  - –Data bytes 2 and 3 contain the identifier for an acknowledge message that the TC1765 sends back to the Host.
  - –Data bytes 4 and 5 make up the 16-bit value for the number of messages to receive

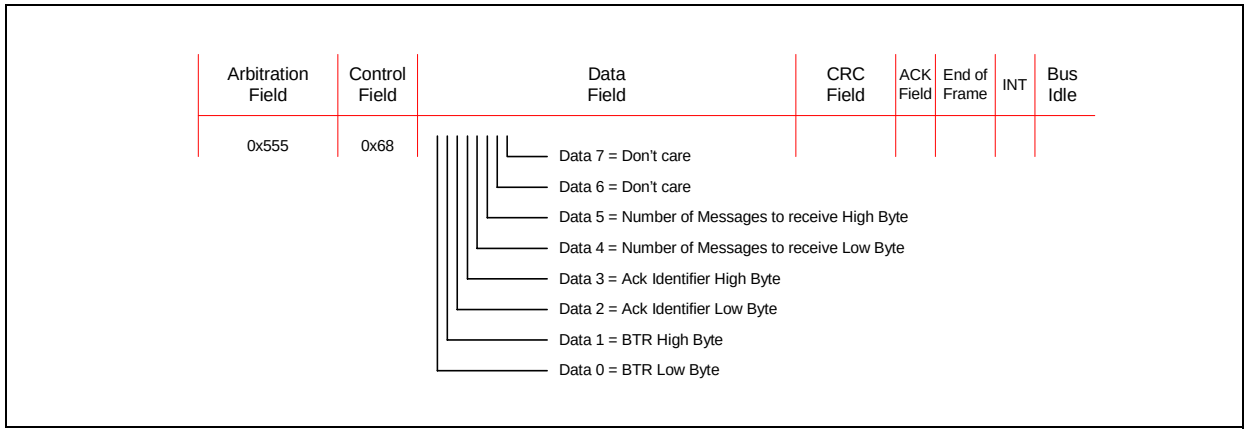


Figure 13 Format of the first message transmitted by the Master

The Host will transmit this message and wait for the TC1765 to acknowledge it. Since there are no other nodes (Point-to-Point) on the bus, the CAN controller of the Host will continually repeat the transmission of the message. This is because the TC1765 has not responded with a dominant bit in the Acknowledge Slot of the transmitted message from the Host. After the first message the Host's CAN error management logic will send error frames until its internal error state changes from error active to error passive. After a short period of time the TC1765 should be able to detect the Host's baud rate and acknowledge the Host's message.

4.6 Message Detection using the Analyzing mode

There is a CAN Analyzer Mode available on the TwinCAN which provides the means to monitor CAN bus traffic without participating in the CAN protocol itself. This mode is used in conjunction with the bit measurement to determine non-destructively (without errors) the baud rate at which the Host is transmitting its message. The TC1765 will attempt to recognize the message being broadcast by the Master in analyzer mode. Once the TC1765 can detect the Master's message without errors, the TC1765 will enable its CAN module. The CAN module on the TC1765 will now acknowledge the Master's message. This signals the Master that communication has been established with the TC1765. If the message cannot be detected within a time-out period, the process will be restarted by recapturing dominant bits and repeating these steps until a message can be found.

4.7 Loading the Code

The first message transmitted by the Host contains three variables necessary to load the code/data. Data bytes 0 and 1 contain the information for the TC1765's Bit Timing Register (BTR). This provides the Host the ability to change these protocol parameters to suit the individual user's needs. However, the user must insure that the correct value is sent; otherwise, communication could be halted. The TC1765 will re-initialize its CAN module to these parameters and transmit a data frame with the identifier (data bytes 2 and 3) sent from the Host.

Data bytes 4 and 5 tells the TC1765 the number of code/data messages to receive. All messages received from this point on will have their data bytes sequentially written into the internal SPRAM starting at location C000 0004_H. Since data bytes 4 and 5 are defined by the Host, the Host has the ability to decide on how large of a program to load. Therefore, the theoretical maximum number of code/data bytes that can be received is equal to 524,280 (65535 * 8 code/data bytes). However, the size of the internal SPRAM is 16 Kbytes which results in a maximum of 8,191 16-bit instructions.

Note: The bootstrap loader assumes all message data is valid. The Host should send its code/data sequentially in multiples of 8 code/data bytes. The user is limited to sending a maximum of 2047 messages.

4.8 Exiting the Bootstrap Loader Mode

Once all messages have been received, the bootstrap loader will transfer program execution to the user code by jumping to location C000 0004_H (i.e. the first loaded instruction). The Bootstrap Loader sequence is now terminated and the program that was loaded from the Host is now executing.

While in bootstrap loader mode, it is also possible to execute normal user mode. This is because the boot ROM is located in its own memory space.

4.9 Choosing the Baud rate for the Bootstrap Loader

When choosing a baud rate, the Master must determine what CAN baud rate is possible for the TC1765 to detect. The major consideration in this determination is the operating frequency of the TC1765. In general, it is recommended to select the slowest possible baud rate for the initial message with a Sample Point (SP) at 80%. Once communication has been established, the baud rate can be changed to a higher rate.

The bootstrap loader software is polling the CAN receive pin to determine the amount of time that is required for a bit to be transmitted. The higher the ratio of CPU frequency to CAN bit time, the greater the likelihood for early baud rate detection.

Once the bit time has been determined as described in **Section 4.4**, the algorithm for determining the Host baud rate is started. The baud rate is determined by performing an iterative loop using the parameters in **Table 7**, along with equations **[2]** and **[3]**. The first objective of the algorithm is to find a solution that uses the maximum number of time quanta (N_{tq}). This is done by looping through the three possibilities in **Table 7** to get the calculated timer value result for each possibility. If an exact match between the calculated timer value and the measured timer value is found, then the loop will exit with the current parameters. However, if an exact match is not found, then a search is made to select the closest calculated value to the measured value. The process continues from this point on as described in **Section 4.6**.

Table 7 Time Quanta Parameters for 80% Sample Point

Item	N_{tq}	Tseg1	Tseg2	SP
1	20	15	4	80.0%
2	15	11	3	80.0%
3	10	7	2	80.0%

The dependency between the system timer ticks and one CAN bit time (one Time Quantum) is calculated with the following equations:

$$N_{tq} = TimerValue = \frac{t_{CANbit}}{f_{SystemClk}} \quad [1]$$

$$BPR = \left(\frac{TimerValue}{N_{tq}}\right) - 1 \quad [2]$$

$$CalTimerValue = (BPR + 1) \bullet N_{tq} \quad [3]$$

$$N_{tq} = Sync + Tseg1 + Tseg2 \quad [4]$$

Where:

BPR	Baud rate prescaler value
TimerValue	Number of System Timer ticks
N_{tq}	Number of time quanta per bit period
t_{CANbit}	One CAN bit time
Tseg1	Number of time quanta before the sample point
Tseg2	Number of time quanta after the sample point
Sync	Synchronization segment (always equal to 1)
$f_{SystemClk}$	CPU operating frequency

From the ISO-DIS 11898 standard, the following assumptions about the bit timing parameters can be made.

$$8 \le t_q \le 25$$

$$3 \le TSeg1 \le 16$$

$$2 \le TSeg2 \le 8$$

$$1 \le SJW \le 4$$

The value for SJW is always assumed to be one.

4.10 TwinCAN Boot Algorithm

Figure 14 shows a flowchart of the TC1765 frequency detection.

Figure 15 shows the Host / TC1765 Bootstrap Loader flow.

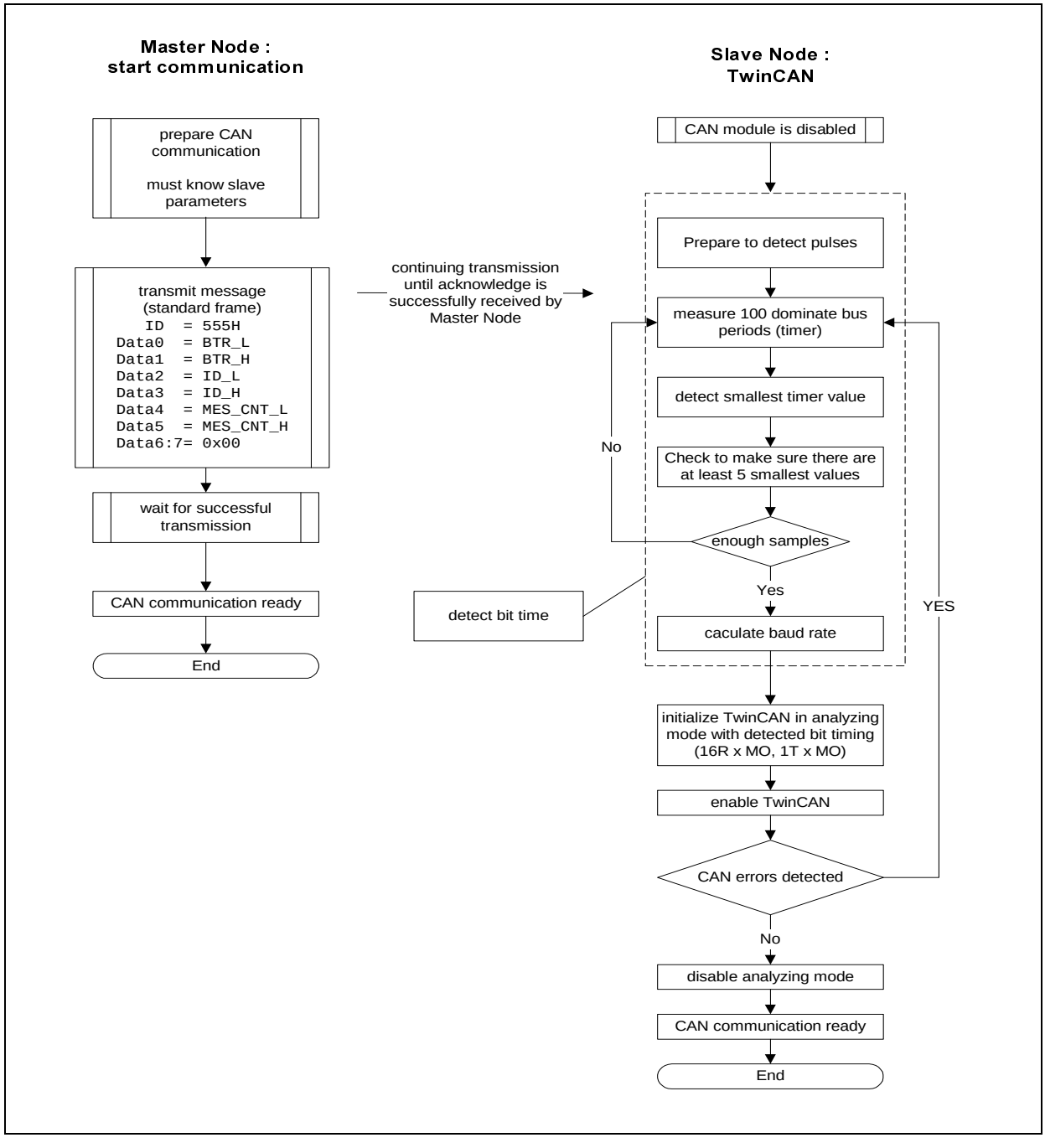


Figure 14 TC1765 Frequency Detection

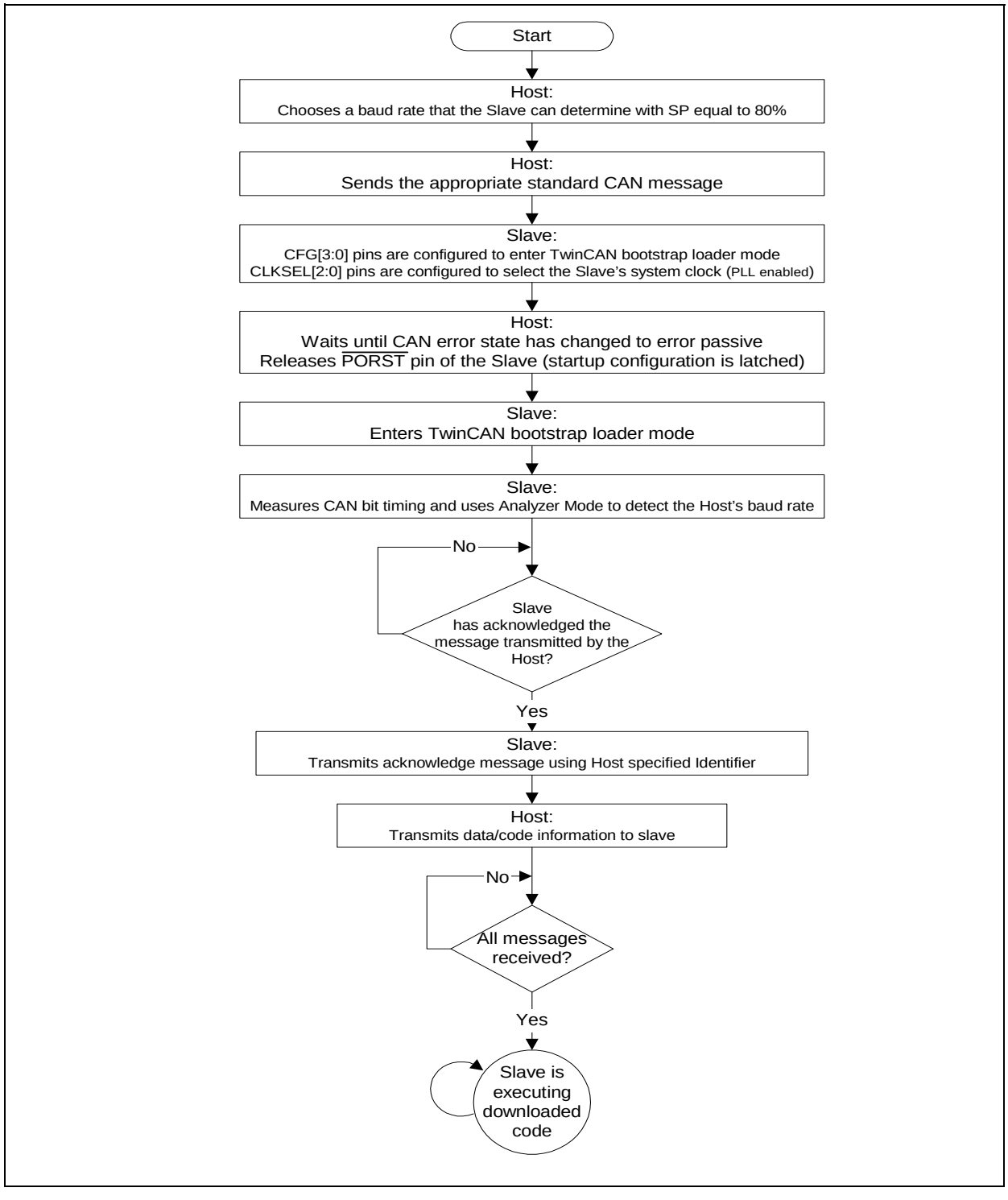
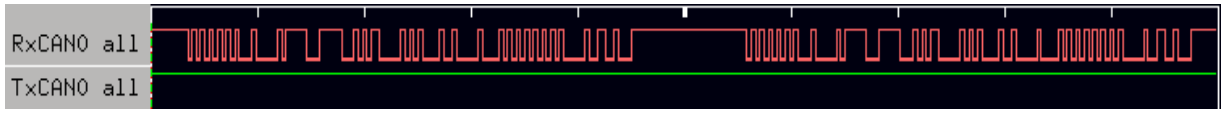


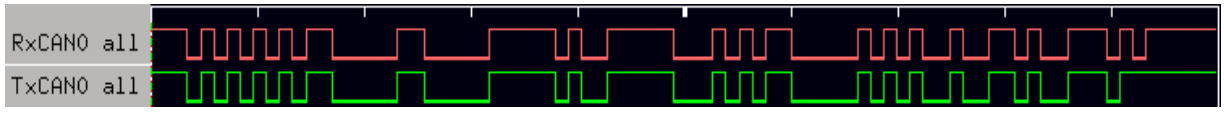
Figure 15 Host Boot Flow

Host message (error active)



Host message (error passive)

TC1765 TwinCAN module acknowledges the host message



TC1765 TwinCAN module transmits response to the host

Overview of the CAN Tx/Rx Sequence

Figure 16 CAN Plots of Data Sequences

Infineon goes for Business Excellence

“Business excellence means intelligent approaches and clearly defined processes, which are both constantly under review and ultimately lead to good operating results.
Better operating results and business excellence mean less idleness and wastefulness for all of us, more professional success, more accurate information, a better overview and, thereby, less frustration and more satisfaction.”

Dr. Ulrich Schumacher

http://www.infineon.com

Published by Infineon Technologies AG