EE389 EDL Report, Electrical Engg Dept. IIT Bombay, Nov 2004

Cryptography with Microcontrollers

Group D15
B. Gopi Vikranth

Supervisor: Prof. P.C. Pandey

Abstract

This EDP aims at designing a low cost device for small message encryption using microcontrollers. The device has the capability to do fast encryption using SHA-1
and to securely protect and rotate secrets. The project can be used in situations where resources are limited to provide strong small-message encryption and peer-to-peer authentication between subsystems.

Contents

1. Introduction:

When systems communicate telemetry or control information between peers the security and authenticity of the communicated data may be important. If the medium is public or could be subject to compromise, securing the communications path becomes an issue. But encrypting control and status messages that are passed between subsystems on networks, telephone lines, or RF channels usually requires extensive microcontroller resources, and the maintenance of secrets (keys) is usually the weak point in the system. Changing or customizing critical system secrets is often impossible in ROM-based equipment, which further reduces the security of the system.

We need low cost devices that contain fast, powerful cryptographic engines. Some of these devices need to have the ability to perform the SHA-1 hash very quickly, and to securely store, protect, and rotate secrets. These devices can be used with small microcontrollers and limited resources to provide strong small-message encryption and peer-to-peer authentication between subsystems.

1.1 Cryptography with Microcontrollers:

When a subsystem component has limited processing power and memory, advanced cryptography is usually not possible. Secure exchange of data and peer-authentication requires secrets, and microcontrollers are not very good at keeping secrets from clever hardware and software attacks. This project will discuss the application of a device that perform SHA-1 hash functions to provide a low-cost, low-overhead cryptographic solution for small message encryption and authentication.

To optimize microcontroller code space, we will use a simple fundamental cryptographic concept called the one time pad. Given an array of bytes that comprise a message, it can be said that the byte-wise XOR (Exclusive-OR) result of that array of bytes against an array of random bytes results in an array of bytes that are equally random. In other words, no information about the message remains in the resulting array. To put it another way, a byte when XOR'ed against a random byte results in an equally random byte. If one was to generate an array of truly random bytes (called a pad), one could then XOR that array against any valid message and the result would be an encrypted message that is unbreakable by any cryptographic means (so long as the pad is held secret and known only to the valid participants in the conversation). This result, when XORed again against the same pad, will be restored to the original message. This is a basic tenet of cryptography—the function is simple (XOR) and the power is entirely in the quality and security of the pad, not in the algorithm by which it is applied to the message.

This would seem to be a very simple, very powerful way to perform message encryption, but there are certain constraints:
1) The pad must be passed from the sender to the recipient in a way that it cannot be compromised. Both parties to the conversation must share the same pad.
2) If the pad is used on more than one message, its strength is greatly diminished—if not wholly compromised. A new pad must be created for each message.

What is needed is the ability to generate an array of bytes (a pad) at will, and then to pass some key with the message that can be used by the recipient to regenerate the same pad. This means that each legitimate participant in the conversation must hold some secret that allows the pad to be regenerated given the key, and the secret must be protected.

The pad generator that we have described is a function called a one-way hash. This function takes input data and generates a digest from it. This digest is entirely affected by every bit of the input data, and yet is derived in such a way that the input data cannot be discovered given the algorithm and the digest.

2 Design:

2.1 The Method :

The following algorithm is being employed

1) The microcontroller generates a random number and sends it to the device.
2) The microcontroller directs the device to generate a SHA-1 digest using the random number and the secret.
3) The microcontroller reads the 160-bit digest from the device.
4) The microcontroller XORs each byte of the message with a byte of the digest (the pad) to obtain the encrypted message.
5) The microcontroller concatenates the random number and the encrypted message and transmits the result to the peer.

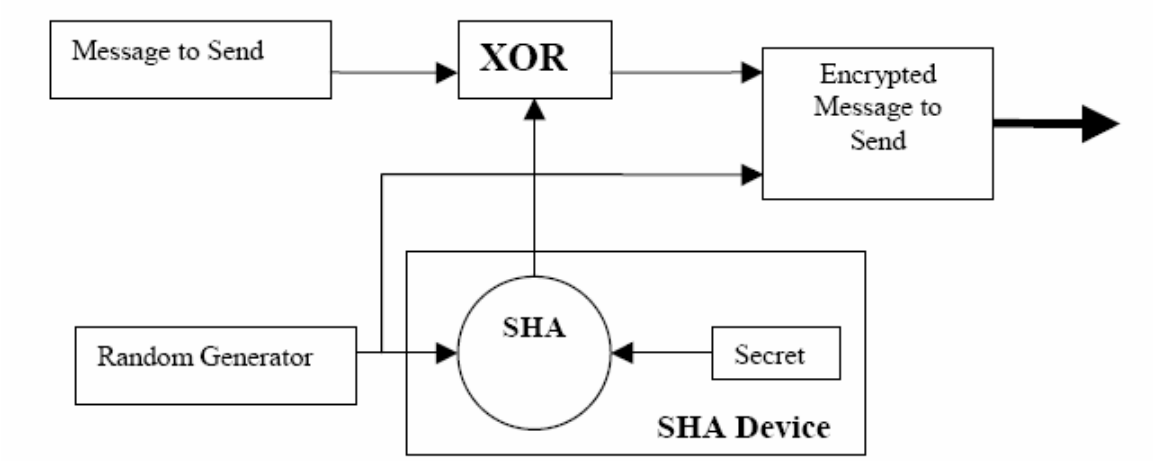


Figure 2.1 Block diagram transmitter

When an encrypted message arrives, the following algorithm is employed:

1) The random number portion of the message is sent to the device.
2) The microcontroller directs the device to generate a SHA-1 digest using the random number and the secret.
3) The microcontroller reads the 160-bit digest from the device.
4) The microcontroller XORs each byte of the message with a byte of the digest (the pad) to obtain the original message.
5) The microcontroller processes the decrypted message.

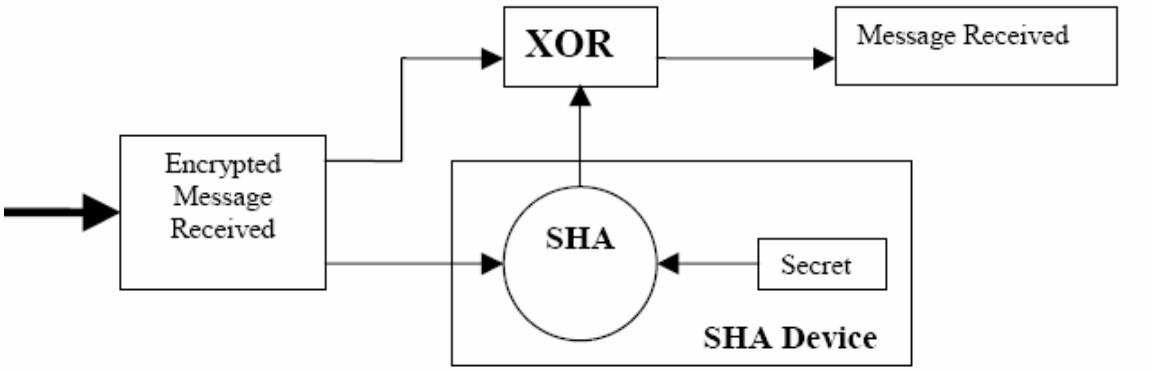


Figure 2.1.2 Block diagram receiver

Security is assured by the strength of the SHA-1 function. Because the SHA-1 hash function is not reversible, the secret cannot be derived from the message traffic. Without the secret, there is no way to decipher or falsify a message. The random seed value used with each message makes every message unique, and makes the deciphering messages all but impossible.

The SHA-1 hash function provides a 160-bit (20-byte) result. When the message to be encrypted exceeds this length, the system may simply perform another SHA operation (to obtain another 20 bytes of pad data).

2.2 SHA Computational algorithm:

This description of the SHA computation is adapted from the Secure Hash Standard SHA-1 document as it can be downloaded from the NIST web site (http://www.itl.nist.gov/fipspubs/fip180-1.htm). The algorithm takes as its input data sixteen 32-bit words M_t ($0 \le t \le 15$), as shown in Tables 1, 2 and 4 for the Compute Next Secret, Copy Scratchpad and Read Authenticated Page command, respectively. The SHA computation involves a sequence of eighty 32-bit words called W_t ($0 \le t \le 79$), a sequence of eighty 32-bit words called K_t ($0 \le t \le 79$), a Boolean function f_t (B, C, D) ($0 \le t \le 79$) with B, C and D being 32-bit words, and three more 32-bit words called A, E and TMP. The operations required for the SHA computation are arithmetic addition without carry ("+"), logical inversion or 1's complement ("\"), EXCLUSIVE OR ("⊕"), logical AND ("∧"), logical OR ("∨"), assignment (":="), and circular shifting within a 32-bit word. The expression "$S^n(X)$" represents a circular shift of X by n positions to the left, with X being a 32-bit word.

The function f_t is defined as follows:

$$f_t(B,C,D) = \begin{cases} (B \wedge C) \vee ((B\backslash) \wedge D) & (0 \le t \le 19) \\ B \oplus C \oplus D & (20 \le t \le 39) \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & (40 \le t \le 59) \\ B \oplus C \oplus D & (60 \le t \le 79) \end{cases}$$

The sequence W_t $(0 \le t \le 79)$ is defined as follows:

$W_t := M_t \quad (0 \le t \le 15)$

$S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \quad (16 \le t \le 79)$

The sequence K_t $(0 \le t \le 79)$ is defined as follows:

$K_t :=$ 5A827999h $(0 \le t \le 19)$

6ED9EBA1h $(20 \le t \le 39)$

8F1BBCDCh $(40 \le t \le 59)$

CA62C1D6h $(60 \le t \le 79)$

The variables A, B, C, D, E are initialized as follows:

A := 67452301h

B := EFCDAB89h

C := 98BADCFEh

D := 10325476h

E := C3D2E1F0h

The 160-bit MAC is the concatenation of A, B, C, D, and E after looping through the following set of computations for t = 0 to 79 (discarding any carry-out):

TMP := $S^5(A) + f_t(B,C,D) + W_t + K_t + E$

E := D

D := C

C := $S^{30}(B)$

B := A

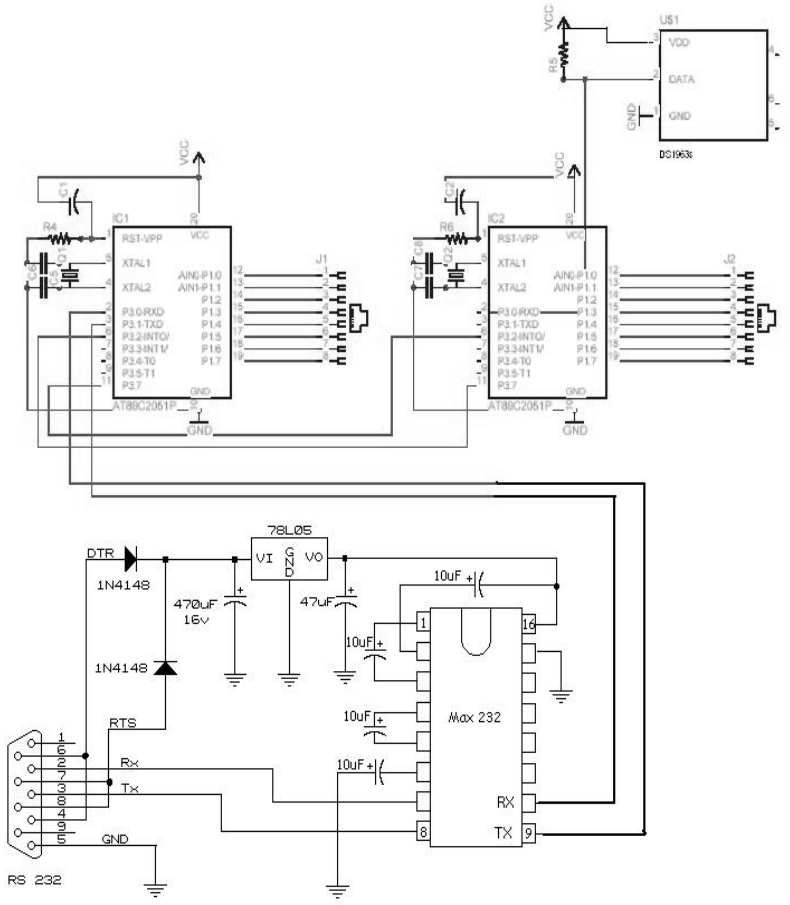
A := TMP

The master can read the Message Authentication Code (MAC) with the Read Authenticated Page command in a register and bit sequence. With the Copy Scratchpad command the bit transmission sequence is the same, however, the master has to compute the MAC and send it to the DS1963S. With the Compute Next Secret command the MAC is not exposed. Instead, the content of the SHA computation registers E and D is directly copied to the secret register.

2.3 1-Wire bus system:

The 1-Wire bus is a system, which has a single bus master and one or more slaves. In all instances the DS1963S is a slave device. The bus master is typically a microcontroller. The discussion of this bus system is broken down into three topics: hardware configuration, transaction sequence, and 1-Wire signaling (signal types and timing). A 1-Wire protocol defines bus transactions in terms of the bus state during specific time slots that are initiated on the falling edge of sync pulses from the bus master.

2.4 Schematic:



2.5 Hardware configuration of the 1-wire interface:

The 1-Wire bus has only a single line by definition; it is important that each device on the bus be able to drive it at the appropriate time. To facilitate this, each device attached to the 1-Wire bus must have open drain or 3-state outputs. The 1-Wire port of the DS2432 is open drain with an internal circuit equivalent to that shown in Figure below multidrop bus consists of a 1-Wire bus with multiple slaves attached.

At regular speed the 1-Wire bus has a maximum data rate of 16.3k bits per second. The speed can be boosted to 142k bits per second by activating the Overdrive Mode. The DS1963S requires a 1-Wire pull-up resistor of maximum 2.2 kΩ for executing any of its memory and SHA function commands at any speed. When communicating with several DS1963S simultaneously, e. g., to install the same secret in several devices, the resistor should be bypassed by a low-impedance pull-up to VPUP while the device transfers data from the scratchpad to the EEPROM and updates the tamper-detect register.

The idle state for the 1-Wire bus is high. If for any reason a transaction needs to be suspended, the bus MUST be left in the idle state if the transaction is to resume. If this does not occur and the bus is left low for more than 16 µs (Overdrive Speed) or more than 120 µs (regular speed), one or more devices on the bus may be reset.

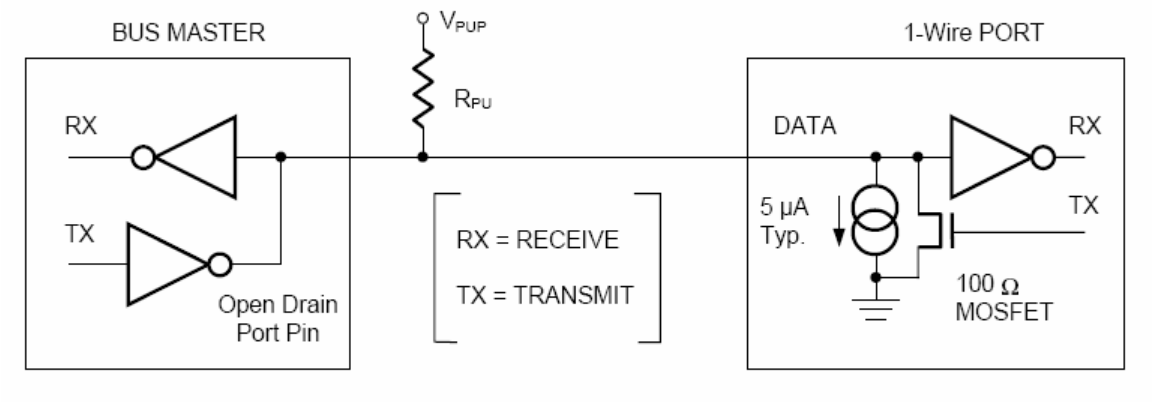


Figure 2.5: Hardware configuration of the 1-Wire device

2.6 Transaction sequence and 1-Wire Commands:

The protocol for accessing the DS1963S via the 1-Wire port is as follows:

- Initialization
- ROM Function Command
- Memory or SHA Function Command
- Transaction/Data

1-wire commands:

- Write Scratchpad Command [0Fh]
- Read Scratchpad Command [AAh]
- Copy Scratchpad [55h]
- Read Memory [F0h]
- Erase Scratchpad [C3h]
- Match Scratchpad [3Ch]
- Read Authenticated Page [A5h]
- Compute SHA [33h]

2.7 RS-232 Protocol

Introduction

In the early 1960s, a standards committee, today known as the Electronic Industries Association, developed a common interface standard for data communications equipment. At that time, data communications was thought to mean digital data exchange between a centrally located mainframe computer and a remote computer terminal, or possibly between two terminals without a computer involved. These devices were linked by telephone voice lines, and consequently required a modem at each end for signal translation. While simple in concept, the many opportunities for data error that occur when transmitting data through an analog channel require a relatively complex design. It was thought that a standard was needed first to ensure reliable communication, and second to enable the interconnection of equipment produced by different manufacturers, thereby fostering the benefits of mass production and competition. From these ideas, the RS232 standard was born. It specified signal voltages, signal timing, signal function, a protocol for information exchange, and mechanical connectors. Over the 40+ years since this standard was developed, the Electronic Industries Association published three modifications, the most recent being the EIA232E standard introduced in 1991. Besides changing the name from RS232 to EIA232, some signal lines were renamed and various new ones were defined, including a shield conductor.

2.7.1 Pin Assignments

If the full EIA232 standard is implemented as defined, the equipment at the far end of the connection is named the DTE device (Data Terminal Equipment, usually a computer or terminal), has a male DB25 connector, and utilizes 22 of the 25 available pins for signals or ground. Equipment at the near end of the connection (the telephone line interface) is named the DCE device (Data Circuit-terminating Equipment, usually a modem), has a female DB25 connector, and utilizes the same 22 available pins for signals and ground. The cable linking DTE and DCE devices is a parallel straight through cable with no cross-overs or self-connects in the connector hoods. If all devices exactly followed this standard, all cables would be identical, and there would be no chance that an incorrectly wired cable could be used. This drawing shows the orientation and connector types for DTE and DCE devices:

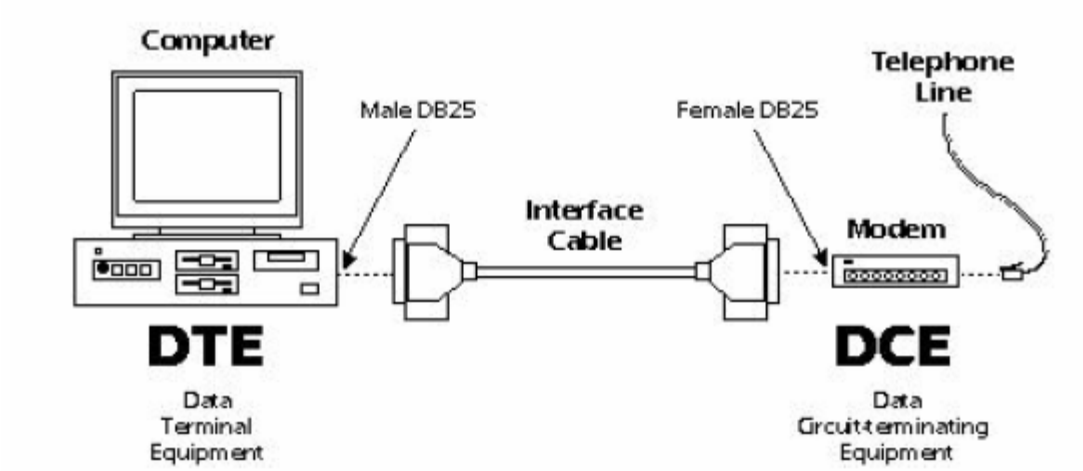


Figure 2.7.1: EIA232 communication function and connector types for a personal computer and modem. DCE devices are sometimes called "Data Communications Equipment" instead of Data Circuit-terminating Equipment.

Here is the full EIA232 signal definition for the DTE device (usually the PC). The most commonly used signals are shown in bold.

Remark:
Signal names that imply a direction, such as Transmit Data and Receive Data, are named from the point of view of the DTE device.

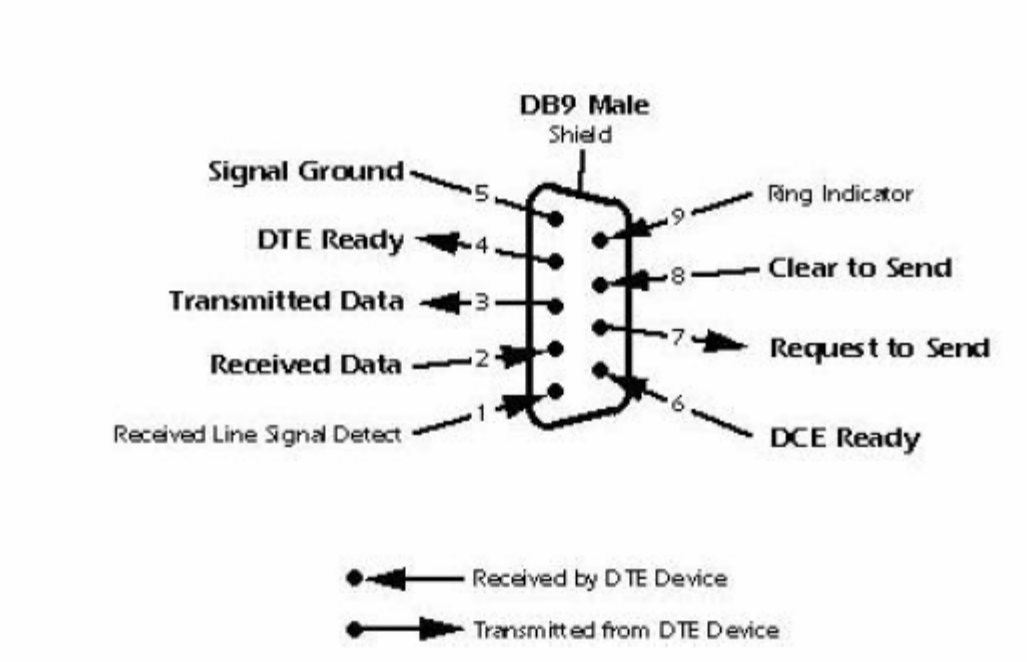


Figure 2.7.2: Pin assignments for db-9 male connector.

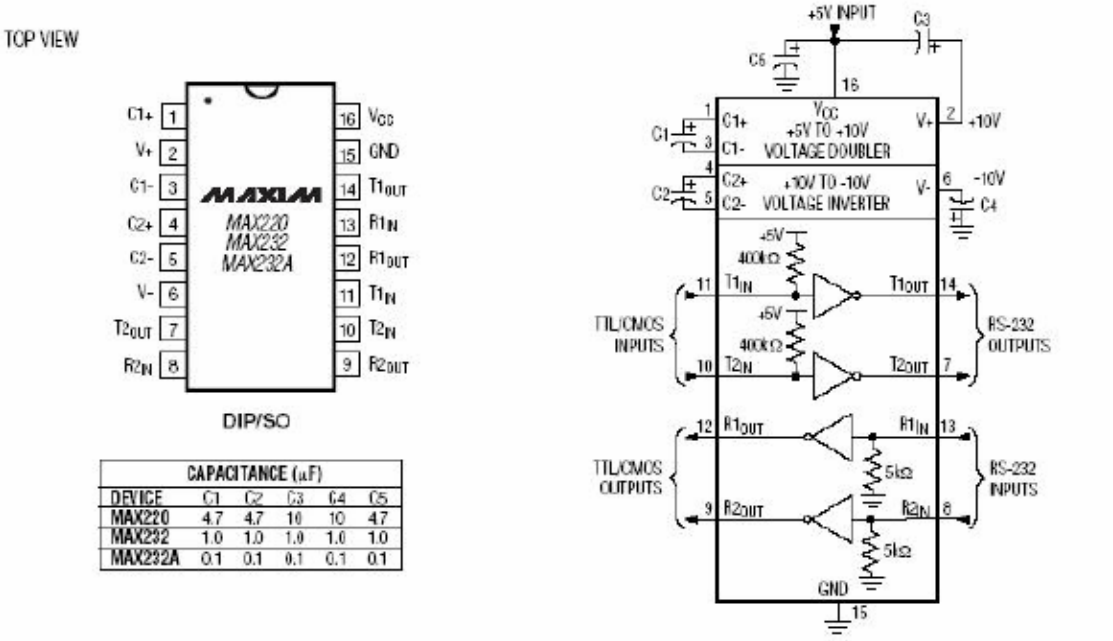
If the EIA232 standard were strictly followed, these signals would have the same name for the same pin number on the DCE side as well. Unfortunately, this is not done in practice by most engineers, probably because no one can keep straight which side is DTE and which is DCE. As a result, direction-sensitive signal names are changed at the DCE side to reflect their drive direction at DCE.

2.7.2 Converting from 232 levels to TTL levels
Signal State Voltage Assignments

Voltages of -3v to -25v with respect to signal ground (pin 7) are considered logic '1' (the marking condition), whereas voltages of +3v to +25v are considered logic '0' (the spacing condition). The range of voltages between -3v and +3v is considered a transition region for which a signal state is not assigned.

2.7.3 MAX232 IC

The MAX220-MAX249 family of line drivers/receivers is intended for all EIA/TIA-232E and V.28/V.24 communications interfaces, particularly applications where 12V is not available.



CAPACITANCE (µF)					
DEVICE	C1	C2	C3	C4	C5
MAX220	4.7	4.7	10	10	4.7
MAX232	1.0	1.0	1.0	1.0	1.0
MAX232A	0.1	0.1	0.1	0.1	0.1

Figure 2.7.3: MAX232 IC

Dual Charge-Pump Voltage Converter:

The MAX232 has two internal charge-pumps that convert +5V to ±10V (unloaded) for RS-232 driver operation. The first converter uses capacitor C1 to double the +5V input to +10V on C3 at the V+ output. The second converter uses capacitor C2 to invert +10V to -10V on C4 at the V- output.

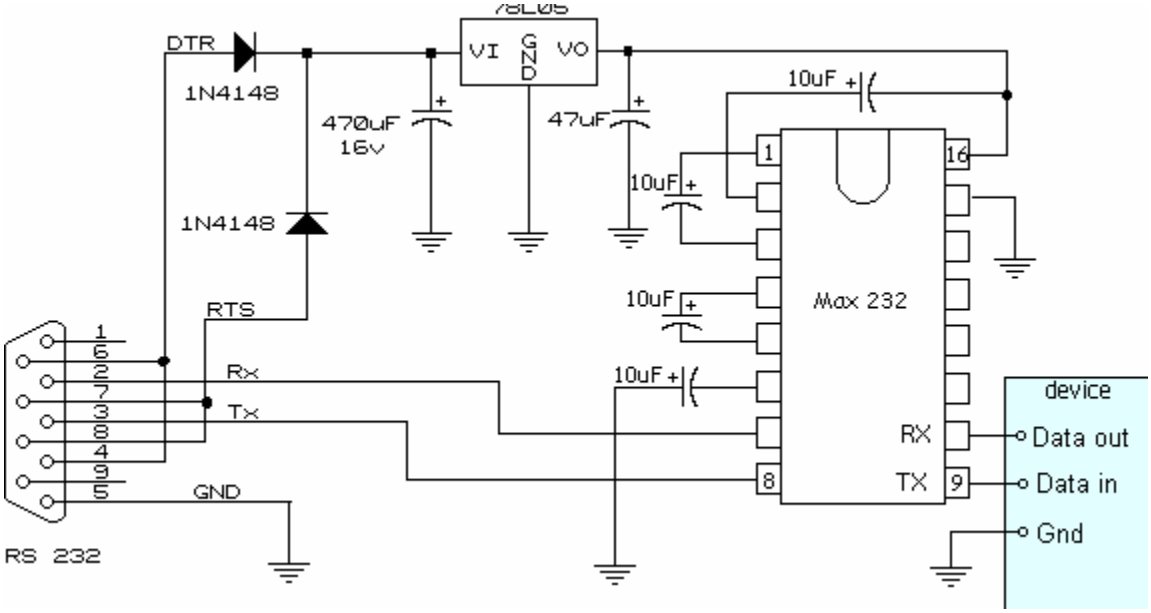


Figure 2.7.4: hardware configuration of RS 232 part

2.7.4 RS-232 Drivers:

The typical driver output voltage swing is ±8V when loaded with a nominal 5k RS-232 receiver and VCC = +5V. Output swing is guaranteed to meet the EIA/TIA- 232E and V.28 specification, which calls for ±5V minimum driver output levels under worst-case conditions. These include a minimum 3k load, VCC = +4.5V, and maximum operating temperature. Unloaded driver output voltage ranges from (V+ -1.3V) to (V- +0.5V).

2.7.5 RS-232 receivers

EIA/TIA-232E and V.28 specifications define a voltage level greater than 3V as a logic 0, so all receivers invert. Input thresholds are set at 0.8V and 2.4V, so receivers respond to TTL level inputs as well as EIA/TIA-232E and V.28 levels. The receiver inputs withstand an input overvoltage up to ±25V and provide input terminating resistors with nominal 5k values. The receivers implement Type 1 interpretation of the fault conditions of V.28 and EIA/TIA-232E. The receiver input hysteresis is typically 0.5V with a guaranteed minimum of 0.2V. This produces clear output transitions with slow-moving input signals, even with moderate amounts of noise and ringing. The receiver propagation delay is typically 600ns and is independent of input swing direction.

3. Cryptanalysis:

Different attack scenarios on this device and the security of the device

Basic Terms:

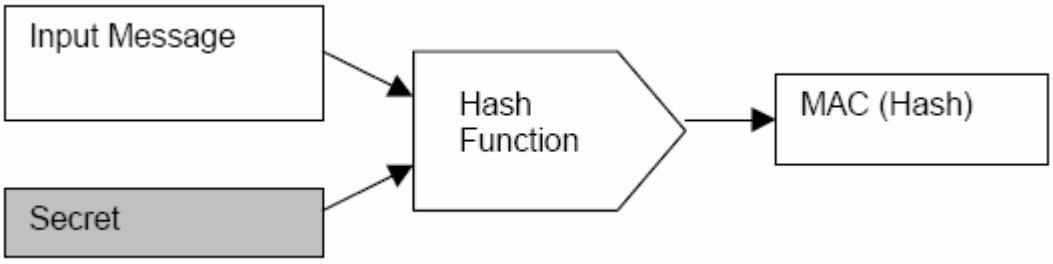
Hash

A hash is a distillation of a message. This distillation is typically much smaller than the message and is a constant size. A *cryptographically strong* hash must be *nonreversible*; meaning that by looking at the hash result there is no way to derive any part of the original message. It must also change significantly with any small change, even a single bit, in the input message. This is called the *avalanche effect*. The hash should also be *collision-resistant,* meaning that it is impractical to find two messages with the same hash. A hash with these properties can be used to verify that a message has not been altered.

MAC

A MAC (message authentication code) is a hash where a portion of the input message is secret. Only participants that know the secret can re-compute and verify the authenticity of the MAC.

MAC generation:



Participants in a system (secret known) can therefore verify a message and MAC combination to be authentic. The transient ones and zeros mentioned earlier are now trusted without resorting to encryption.

SHA-1 takes in one or more blocks of 512 bits (64 bytes) and creates a 160-bit (20-byte) hash.

Copy Attack

The *Copy Attack* is done by copying valid service data from a device that is part of the service and writing it to another device that may or may not be part of the service. A device is part of the service if it has the correct authentication secret installed. The purpose of this attack is to take a valid token and create another one from it, thus creating

two valid tokens. This could also be done to copy the attributes such as identity or monetary value to another device to create more money or give access where none was provided.

Conditions	How the attack is defeated
Target device is *not* part of system the (authentication secret *not* set)	Since the authentication secret of the target device is not set to the correct value, then the device will not be accepted by the service no matter what the data is.
Target device is part of the system (authentication secret set correctly)	The target device will pass the authentication test. However there is a second MAC (message authentication code) which is embedded in the service data. This MAC (*Service Data Signature*) has the token's *ROM ID* as part of the SHA-1 calculation. Since the target device has a different *ROM ID*, then the data will be considered invalid and the device will be rejected.

Replay Attack

The *Replay Attack* attacker makes a copy of the service data of a valid device. The attacker then uses up other usage fields. The saved data is then copied back to the device. This way, the old data is *replayed*.

Conditions	How the attack is defeated
None	Each page used for secure data has a read-only non-rollingover page write-cycle counter associated with it. While the device still has a correct secret to generate the authentication MAC, the second MAC (*Service Data Signature*) that is embedded in the service data will be invalid. This MAC includes the write-cycle counter and a page number that it resides on. Writing old data back into that page or another page will invalidate it. Even rewriting the same data will invalidate it since the write-cycle counter advances.
None	The target device has a write protection feature that requires a valid MAC to copy data to it. Since the attacker does not know the secret, copying to the device is not possible.

Eavesdrop Attack

The *Eavesdrop Attack* is a technique where the communication is monitored to reveal the secret or a repeating pattern that could be replicated.

Conditions	How the attack is defeated
Monitor and log communication during a valid transaction	When authenticating a device to verify whether it is part of the service, a random challenge is presented to be included in the MAC calculation. This random value changes the communication traffic significantly with each transaction. Also, at no time is the secret ever sent over the device except when installing the secrets in a protected environment.

A-B-A Attack

The *A-B-A Attack* tries to play two different monetary SCU (service control unit) off each other to fool them into giving more product then was paid for. This attack starts with the token being presented to the first SCU 'A'. It is then removed before the debit can be performed but close enough to the end so that the SCU thinks it was complete but without verification. This could be accomplished by monitoring the communication and interrupting the sequence at the appropriate time. SCU 'A' will then wait for the token to be reintroduced to verify the debit was complete. The token is then presented to another SCU 'B' and a complete debit is performed with product produced. The token is then taken back to the first SCU 'A' and it is allowed to verify that the transaction was complete. This fools 'A' into thinking it did the debit and product is given again. Consequently, two products were delivered for only one debit.

Conditions	How the attack is defeated
The transaction taking place on SCU 'A' must be stopped at the critical time.	A random value called a *Transaction ID* is part of the service data to make every monetary instance unique. When SCU 'A' goes back to verify that it's debit was complete, it will fail due to an incorrect *Transaction ID*.

Emulation Attack

Use a microprocessor to emulate the behavior of the token. The emulator must be fast enough to respond to the master as if it was a real device. As shown below there is no risk as long as the attacker does not know the authentication secret. This risk is further reduced by the technique of making each authentication secret unique by including the ROM ID as a component in the calculation of the secret (*Unique Authentication Secret*).

Conditions	How the attack is defeated
Authentication secret is not known.	The emulated device can not create the correct MAC for a given challenge so would not be accepted as part of the service.
Authentication secret is known.	None Communicate with the SHA device at the fastest possible data rate. The master (SCU) will expect a SHA computation in 1ms to 2ms. This makes creating an emulator very challenging. Never disclose or expose the authentication secret.

Brute Force Attack (MAC)

A *Brute Force Attack* is an inelegant enumeration of all possibilities to get a desired result. The MAC variation of this attack consists of enumerating through all of the challenge-response pairs and keeping the results in a database to be used by an emulator. This requires access to a valid device that is part of the service. The random challenge is three bytes long giving the number of possible challenge response pairs at over 1.6 million. Since the resulting SHA-1 MAC is 20 bytes long, the data created from this would be: 0xFFFFFF (challenge) * 20 (size of MAC) = 335,544,300 bytes (320MB). This creates a pre-computed MAC look-up table that could be used by an emulator.

Conditions	How the attack is defeated
Dynamic data. Each time the device is presented, a new service record with a *Service Data Signature* is created and written	The new service record must be reread and authenticated after it is written to the token. Since there is a different MAC for authentication of the new service record, the emulator would have to know this also. The service record should contain a random value called a *Transaction ID* so the emulator will not be able to predict what the new service record will be so it could not brute force this new MAC.
Static data. Data is not changed, it is only authenticated.	After authentication of the static data, write random data to an unused page and do another authentication from that page. The emulator will not know this new MAC.

Brute Force Attack (Secret)

The *Secret Brute Force Attack* enumerates through all possible secrets until a correct MAC is produced. A token that is part of the service supplies what a correct MAC is. If the secret is found it could be used in an emulator as long as it has a SHA-1 coprocessor

that could do the hash in a millisecond. The secret is eight bytes long (FFFFFFFFFFFFFFFF hex). If it takes one microsecond to compute SHA-1 on a fast computer then it would take 580 thousand years to enumerate through all of the secrets.

Conditions	How the attack is defeated
Computing power and a virtually unlimited amount of time.	Using unique secrets (*Unique Authentication Secret*) in each device mitigates the severity of the problem. This is accomplished by using the devices unique *ROM ID* as part of the secret generation. If the secret is discovered the system does not have a 'class break'. Only the one device with that particular *ROM ID* can be emulated. Putting that ROM ID on a black-list will prevent its use forcing the attacker to do another brute force attack on a different device.

Microprobe Physical Attack (Secret)

The physical attack is an attempt to probe the internal silicon chip to read the *Unique Authorization Secret*. This is very difficult to do, the EEPROM devices. Usually losing contact with the battery will erase the onboard secrets. As with the brute force attack, if the secret is discovered it could be used in an emulator to make what appears to be a valid device in the service. This emulated device could then be used to replay previous data. The conditions and partial solution is the same for this attack as the brute force secret attack above.

Host/SCU Attack (Secret)

Take the SCU that does the authentication and probe it for the *Master Authentication Secret* and *Master Signing Secret.*

Conditions	How the attack is defeated
Must have physical access to the SCU.	Keep the SCU secure. For example, use a secure Microprocessor.
Must have physical access to the SCU. .	A partial solution is to use a 'coprocessor' for the device authentication and data validation in the SCU. The secrets are safe from detection. However this coprocessor could be used to create valid *Service Data Signatures* if the padding data (*Initial Signature*) is known.

Competitor Attack

The *Competitor Attack* is an attempt to discredit a system by maliciously and intentionally destroying or disrupting data.

Conditions	How the attack is defeated
Token must be presented to a malicious reader.	Any data can be overwritten which could invalidate the token. However, writes to a secret can be detected by checking the write-cycle counter associated with it. The device can be reused if it is reloaded with the correct data and secrets.
Token must be presented to a malicious reader.	The target device has a protection feature that requires a valid MAC to write data to the user memory area. Since the attacker does not know the secret, writing to the device is not possible. The target device also has a write protect mechanism for the secret. To prevent the attacker from overwriting the secret, this feature must be enabled.

4. Testing:

4.1 1-WIRE SIGNALING

The DS1963S requires strict protocols to ensure data integrity. The protocol consists of four types of signaling on one line: Reset Sequence with Reset Pulse and Presence Pulse, Write 0, Write 1, and Read Data. Except for the presence pulse the bus master initiates all these signals. The DS1963S can communicate at two different speeds: standard speed and Overdrive speed. If not explicitly set into the Overdrive mode, the DS1963S will communicate at standard speed. While in Overdrive Mode the fast timing applies to all waveforms.

To get from idle to active, the voltage on the 1-Wire line needs to fall from V_{PUP} below the threshold V_{TL}. To get from active to idle, the voltage needs to rise from V_{ILMAX} past the threshold V_{TH}. The voltage V_{ILMAX} is relevant for the DS1963S when determining a logical level, but not for triggering any events. The initialization sequence required to begin any communication with the DS1963S is shown in Figure below. A Reset Pulse followed by a Presence Pulse indicates the DS1963S is ready to receive data, given the correct ROM and memory function command. In a mixed population network, the reset low time t_{RSTL} needs to be long enough for the slowest 1-Wire slave device to recognize it as a reset pulse. This duration is 480µs at standard speed and 48µs at Overdrive speed. After the bus master has released the line it goes into receive mode (RX). Now, the 1-Wire bus is pulled to V_{PUP} via the pullup resistor.

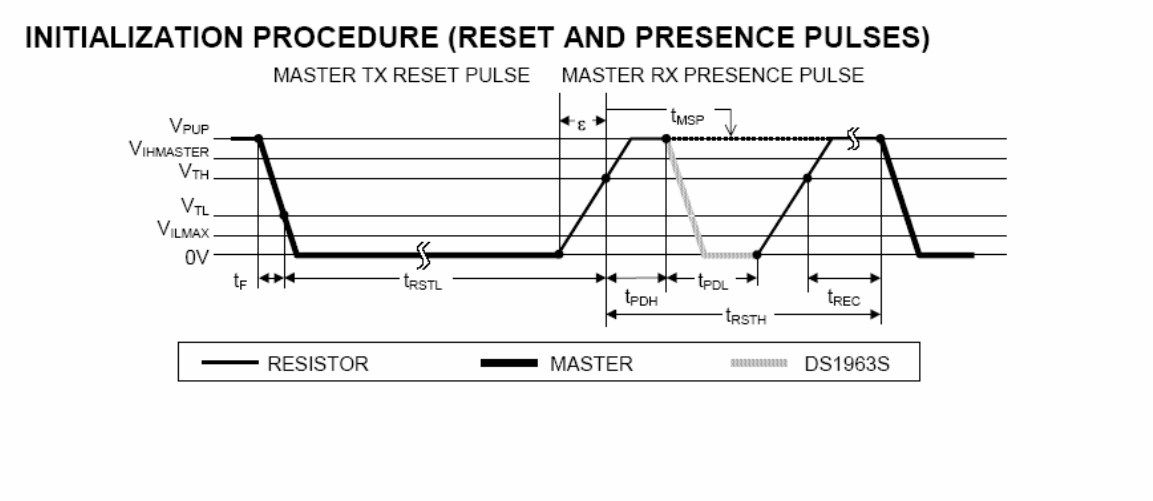


Figure 4.1: Initialization procedure

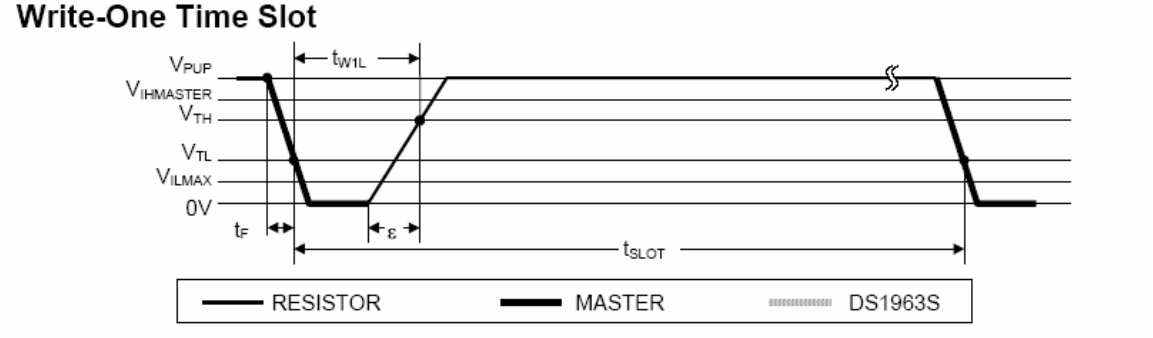
4.2 Read/Write Time Slots

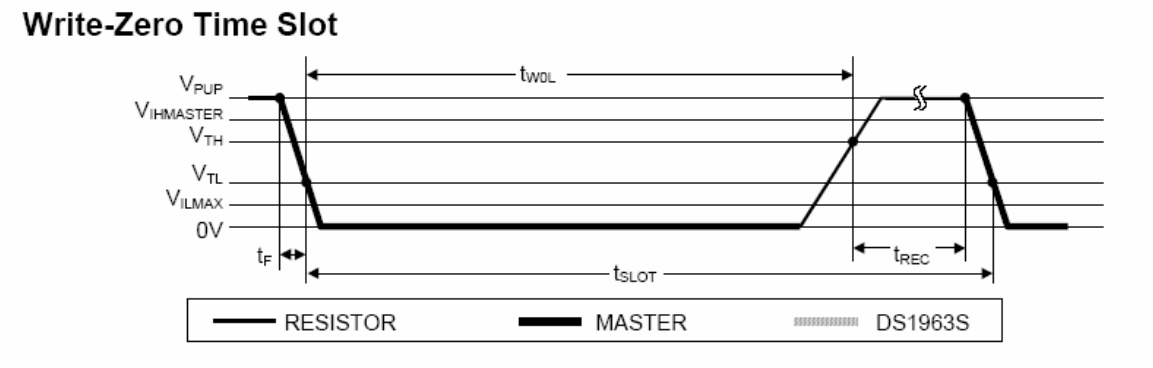
Data communication with the DS1963S takes place in time slots that carry a single bit each. Write time slots transport data from bus master to slave. Read time-slots transfer data from slave to master. The definitions of the write and read time slots are illustrated in Figure below. All communication begins with the master pulling the data line low. As the voltage on the 1-Wire line falls below the threshold V_{TL}, the DS1963S starts its internal timing generator that determines when the data line will be sampled during a write time slot and how long data will be valid during a read time slot.

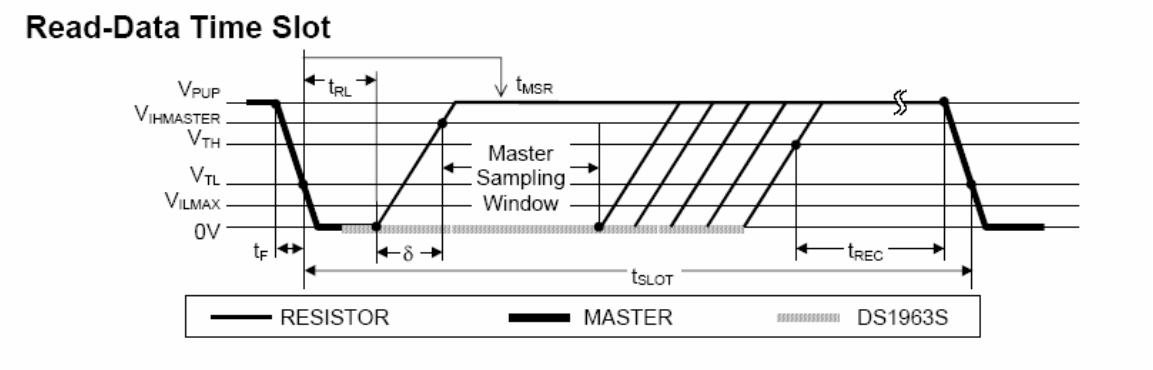
Master to Slave

For a **write-one** time slot, the voltage on the data line must have crossed the V_{THMAX} threshold after the write-one low time is expired. For a **write-zero** time slot, the voltage on the data line must stay below the threshold until the write-zero low time t_{W0LMIN} is expired.

READ/WRITE TIMING DIAGRAM







Slave to Master

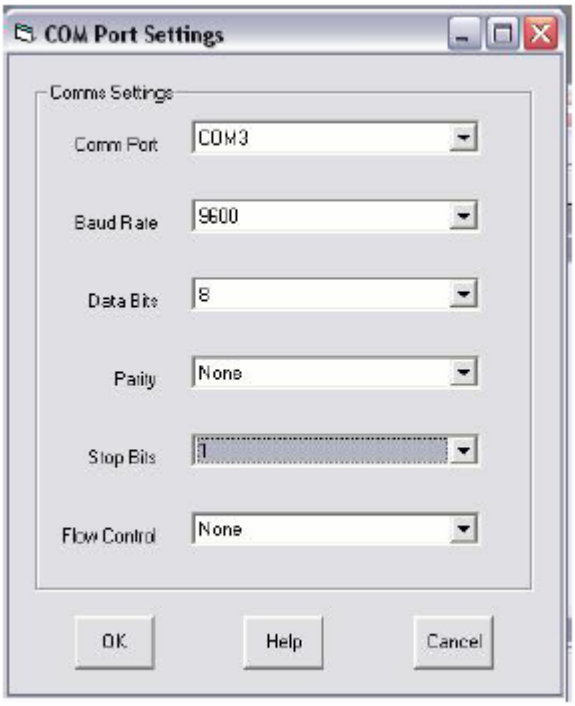
A **read-data** time slot begins like a write-one time slot. The voltage on the data line must remain below V_{TLMIN} until the read low time t_{RL} is expired. During the t_{RL} window, when responding with a 0, the DS1963S will start pulling the data line low; its internal timing generator determines when this pull-down ends and the voltage starts rising again.

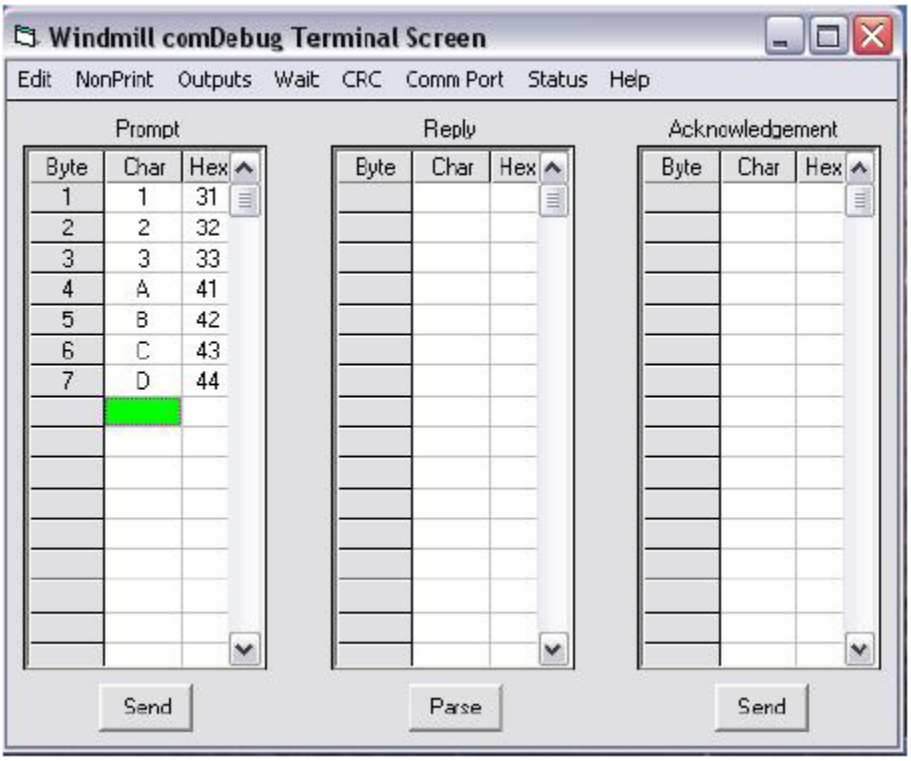
4.3 Windmill comDebug Software

Using comDebug software

4.3.1 Entering Communication settings

- Comm Port: Select the port to which the instrument is connected.
- Baud Rate: The baud rate must be the same as that used by the instrument. Check the instruments Manual for details of which rate to choose. (9600)
- Stop Bits: May be 1 or 2-if in doubt start with 2. 1 was used.





4.3.2 Using Prompt Grid:

Use this grid to enter commands to send to the instrument. The Byte column numbers each byte sent. The Char column shows the printed value of the byte. The Hex column

shows its equivalent Hex value. One can type either into the Char or the Hex column. Click the cell where data has to be entered: it turns green. Simply type the characters to be sent.

4.3.3 Checking RS-232 Port

I connected the db-9 connector to the COM 3/4 port on the PC and connected the RXD, TXD and GND lines appropriately. Then to test the system, a simple code in the Microcontroller that sends back all data received was used.

5. Applications

Authentication is the process of proving to a host that a device, person, or message is valid and authentic. The authentication technique that takes advantage of a MAC is called *challenge and response*. A challenge is random data that is presented by a host to the device to be authenticated. The challenge is then included in the MAC calculation. This allows for every authentication session to be different and non-deterministic.

What can be authenticated? A message between two pieces of equipment could be authenticated if the secret is known by both pieces. A portable token could be authenticated to a door to allow access to a controlled room. See Table below for a list of potential applications including both the host and authentication target.

Table 1. APPLICATIONS USING MAC

Application	Host	Authentication Target
Device authentication	Equipment	Peripheral device with embedded token
Physical access control	Electronic door lock Building access Filing cabinet lock Safe	Portable token
User access control	Workstation	Portable token and user
Software authorization	PC Software Equipment Firmware	Portable, attached, or embedded token
Electronic cash (eCash)	Vending machine Parking meter Toll booth Pay phone Gaming	Portable token

Status:

- The circuit was tested on the bread board.
- Boxed PCB is made.
- The device is working for a baud rate of 9600 bps. Due to the unavailability of the smd device further testing is pending.

6. Future Scope and conclusions:

This device can do small message encryption with microcontrollers. This device can be used to provide strong encryption and authentication of control and status messages, telemetry, or sensitive process control information. For low cost and low overhead, it provides nonvolatile memory, secure secret storage, secret sharing and rotation, fast SHA-1 pad generation, and a globally unique serial number.

A few features that can be incorporated in the design to enhance its capability such as:

- Using a microcontroller with two RS 232 ports for optimizing the resources.
- Providing external interrupts to actually monitor the data and providing data taps so that testing can be done more easily.
- Error correcting code algorithms
- Inclusion of port for Infrared transmit/receive data.

Acknowledgement:
I would like to take this opportunity to thank **Prof. P. C. Pandey** and **Prof. V. R. Sule** for their constant assistance and guidance.

References:

Secure Hash Standard
Federal Information Publication Standard (FIPS) specifying the SHA-1 standard.
Document FIPS 180-1 (HTLM)
URL http://www.itl.nist.gov/fipspubs/fip180-1.htm
Applied Cryptography, Second Edition
Good general introduction text on cryptography techniques and theory. One way hash functions including SHA-1 are described in Chapter 18.
Document Bruce Schneier, John Wiley & Sons, 1996 (Print)
URL http://www.wiley.com/cda/product/0,,0471128457|desc|2941,00.html
Handbook of Applied Cryptography
Extremely detailed text covering various cryptographic techniques including hashes. Chapter 9 specifically covers hashes and data security.
Document A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996 (Print, PS, PDF) URL http://www.cacr.math.uwaterloo.ca/hac/
Information technology – Security techniques – Hash-Functions – Part 3
ISO standard presenting various hash standards including SHA-1. This standard is then referenced in other ISO standards relating to MAC creation.
Document ISO/IEC 10118-3 Standard (PDF for purchase)
URL http://webstore.ansi.org/ansidocstore/product.asp?sku=ISO%2FIEC+10118%2D3%3A1998