DS87000
Microcontroller Programmer

www.dalsemi.com

Features

- Operates in Stand Alone or PC-Hosted Modes
- Easy to use pull-down menus in PC-Hosted mode
- Operation from front panel in stand alone mode
- Uses efficient "intelligent" programming techniques
- Non-volatile data and operating parameter storage
- Optional automatic device serialization (unique identifier written to every device programmed)
- Programs 40-pin DIP packages; DS87001 plug-in adapters available for other package types
- Includes 110 volt (DS87000-000 for U.S.) or 220 volt (DS87000-220) power supply

Description

The DS87000 is a programming solution that provides a convenient, inexpensive method of programming Dallas Semiconductor's family of EPROM-based Microcontrollers. The system comes complete with the programmer, menu based PC software, power supply, and RS-232 cable. The programmer may be operated in a stand-alone or PC hosted mode. Initially, a PC loads programming information into the DS87000. Once loaded, there is no further requirement for the PC and the programmer may then be operated completely from its front panel if desired. This allows the DS87000 to be easily moved to where it is needed. When a PC is available, the DS87000 may be operated from the menu-based software provided.

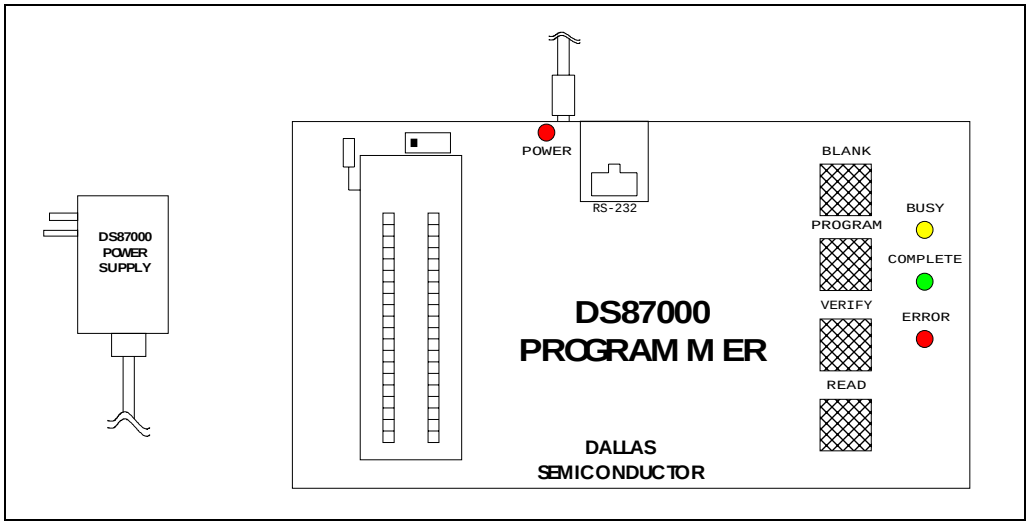
Operations in front panel in stand-alone mode are initiated via the programmer's four push buttons. Three lights (LEDs) provide feedback on system status. The convenient plug-in transformer provided with the programmer supplies all necessary power.

When operated in PC-hosted mode, the user is offered full flexibility of the programmer through a system of pull-down menus. All commands, options and menus are arranged to provide convenient access to each command. The main menu screen is divided into DOS, Buffer, Encryption, Device, and Script Definition menus. Each of these menus contains sub- for controlling various parts of the system.

The DS87000 also offers a unique feature typically found only in more expensive programming equipment. This feature allows the DS87000 to write a unique identification number to every device programmed. After initial conditions are established, the programmer automatically writes and subsequently increments a serial number unique to each device.

The following sections describe operational details and installation procedures of the DS87000. The overall programming environment is comprised of the ***programmer***, the device to be programmed (or simply the ***device***), the ***system software***, and the optional standard personal computer (***PC***).

DS87000 HARDWARE: Figure 1



System Software

The DS87000's system software runs on a standard PC, and allows full control of the programmer's operation through a system of pull down menus. Each of these menus and their sub-menus is described in the following sections.

DOS Menu

Port

The Port command allows the user to select which PC COM port (1, 2, 3, or 4) the programmer is attached. The power-on default is COM 1.

DOS

This function provides a convenient means of temporarily exiting to a DOS shell. The DOS "EXIT" command will return to the menu.

Exit

Exit the DS87000 system software.

Buffer Menu

Load

This command loads an Intel hex file into the programmer's data buffer. Only the buffer memory that is loaded with data is affected. All other memory remains unchanged.

Save

This command saves the contents of the programmer's buffer to the selected PC file. The information is stored in Intel hex format. The size of the EPROM in the last device read or programmed determines the file size. All buffer memory used for a particular device is saved to the file. Therefore if the device most recently read or programmed has a 16K Byte EPROM, the file will contain the Intel hex formatted contents of all 16K bytes.

Drive
This command allows the user to select another disk drive for the files used in the buffer menu's load and save commands.

Compare
This command compares the contents of the buffer with the contents of the selected file. Every byte contained in the file is compared to the corresponding byte (same device address) in the buffer.

Fill
Fill the programmer's buffer with a specific value or a random number. When unsure of the buffer memory contents, this command can be used to initialize the buffer to a known value. Also, by using the random option to fill the buffer with random numbers before loading a program, the encryption function will provide more security. The user is prompted to enter the beginning and ending address of the range to be filled. The default range is displayed, and may be modified or accepted as is. The user is also prompted to enter a value (in hex) to use for filling. Entering an "R" will cause a series of random numbers to be used.

Edit
This command provides full screen edit capability of the programmer's data buffer memory. The cursor keys allow character by character movement within the buffer. In addition, page up, page down, home, end, and their control key combinations allow movement in larger increments.

The information contained in the buffer memory is presented in hex format in the center of the screen, and the ASCII representation on the right. If the hex value cannot be represented in printable ASCII, a period (.) is displayed. The cursor may be placed over either the hex or ASCII value and a new value entered. When exiting the edit function after modifying one or more bytes of the buffer, the user will be asked if the changes should be saved.

Encryption Menu

Load
Load an encryption vector from a PC file to the programmer. The PC file is a standard ASCII file representing hex values, and may be created using any ASCII text editor. The file must include exactly 64 hex values (separated by spaces), which will be read and stored sequentially. During a load, if the file does not contain exactly 64 hex values, the programmer's encryption vector will not be modified, and an error message will be displayed. Note that this command simply loads an encryption vector into the programmer. It is the Write Encryption Vector command of the Device menu that actually stores the vector into the device.

Save
Save the current encryption vector contained in the programmer to a PC file. This information is stored as standard ASCII representing 64 hex values.

Drive
This command allows the user to select another disk drive for the files used in the encryption menu's load and save commands.

Edit

This command provides full screen edit capability of the programmer's encryption vector. All bytes of the vector are displayed on the screen, and each can be individually edited. The vector is presented in hex format in the center of the screen, and the ASCII representation is to the right. If the hex value cannot be represented in printable ASCII, a period (.) is displayed. The cursor may be placed over either the hex or ASCII value and a new value entered. When exiting the edit function after modifying one or more bytes of the vector, the user will be asked if the changes are to be saved. If yes, the new vector will be stored in the programmer.

The user may clear the entire encryption vector by entering an ALT-C. Note that a cleared encryption vector is all ones, and if a cleared vector is stored in the device, there will be no encryption. A series of random numbers will be used to fill the encryption vector if an ALT-R is entered. A randomly selected encryption vector provides a more secure device.

DEVICE MENU

Blank Check

This command reads the device to confirm that it is completely erased (all 1's). The user is prompted to enter the beginning and ending address of the range to be checked. The default range is displayed, and may be modified or accepted as is.

Read

This command will read all or part of the device's EPROM and store it in the programmer's buffer memory. The user is prompted to enter the beginning and ending address of the range to be read. The default range is displayed, and may be modified or accepted as is.

If an encryption vector has been stored in the device, the data will be encrypted as it is read. If the intent is to read non-encrypted data, the read must be performed before the encryption array is written to the device. The lock bits also have an effect on this function. If either of the two highest levels of lock are written to the device, the data read will be all ones.

Verify

Verify that the device's EPROM contains the same information as the programmer's buffer. The user is prompted to enter the beginning and ending address of the range to be verified. The default range is displayed, and may be modified or accepted as is.

Since the verify function actually reads data from the device, verify must be performed before the device's encryption array, lock bits, and/or serial number are written. After the encryption array is written to the device, all data that is read will be encrypted as a standard function of this security measure. After the lock bits are written, it may not be possible to read the device at all, depending on the level of lock. The serial number is unique for every device programmed, so it will not verify from one device to another.

IBlank Check

This command reads the device to confirm that it can be programmed with the contents of the buffer. This mode makes use of the fact that a completely blank device is not required for proper programming. For instance, if the device contains a 0 bit where a 0 bit is to be programmed, then it does not matter that the bit is not "blank" (i.e., 1). Also, if a byte already contains the byte to be programmed, then it does not need to be re-programmed, and it does not matter that it is not blank. The user is prompted to enter the beginning and

ending address of the range to be checked. The default range is displayed, and may be modified or accepted as is.

Program

Program the device's EPROM memory with the contents of the programmer's buffer. The user is prompted to enter the beginning and ending address of the range to be programmed. The default range is displayed, and may be modified or accepted as is.

Unlike the program button, the program command only writes data from the buffer to the device. It does not perform multiple functions as the button does (see hardware section). If operations other than programming are desired, the Program Via Script command must be used.

Serialize

This command allows the user to store a unique serial number into each device programmed. The user is prompted to enter hex values for the parameters Length, Address, and Value. The Length parameter determines how many bytes the serial number will occupy (up to a maximum of 4 bytes = 32 bits). The Address parameter determines the device's EPROM address where the least significant (or only) byte of the serial number will be stored. Subsequent bytes will be stored at higher addresses. The Value parameter sets the initial serial number (that is, the next number to be programmed). The user must ensure that the area of EPROM that the serial number will be written to is blank and unused by the data. As this menu is exited, the user is allowed to write the serial number to the device immediately or simply store the changes to the parameters.

After programming the serial number into a device, it is treated as a simple binary number and incremented by one. In the event that it increments beyond what can be represented in the user defined Length parameter, the serial number wraps around to 0. Therefore, this parameter must be chosen to accommodate the maximum number of devices that will be serialized.

All of the command parameters above are stored in non-volatile memory in the programmer, so they will continue to be in effect even after power is removed. Since serializing writes a unique number to the device’s EPROM, this function must be performed after the device is verified (unless a verify over an appropriate range is performed). Otherwise an error will be detected.

Option Byte

This command allows the user to write the contents of the device's option byte. For specific option byte bit definitions, refer to the device's data sheet. Unused bits should be set to 1’s.

Write Encryption

Write the encryption vector contained in the programmer's memory into the device. Note that after this operation, read and verify device operations will produce encrypted output if the encryption vector is non-blank. If the encryption vector is blank (all 1’s), the encryption has no effect.

Write Lock Bits

This command allows the user to write the desired lock bits into the device. The bit settings and therefore the degree of lock security is dependent on user input. Since level 1 lock is the device’s default condition, this level is not an option on the menu. Each higher level of lock includes the security of all lower levels as indicated below.

- **Level 1:**

There is no lock. However, encryption will be performed as usual. This level is the erased default state of the device. The programmer does not program this level.

- **Level 2:**
  This level prevents MOVC instructions in external memory from reading program bytes in internal memory. \EA is sampled and latched on reset. No further programming of EPROM will be allowed after this level is set.

- **Level 3:**
  This is the same as level 2 plus it does not allow further verify or read operations. It also prevents MOVX instructions in external memory from reading SRAM (MOVX) in internal memory.

- **Level 4:**
  This is the same as level 3 plus it does not allow execution from external memory.

Note:

The lock bits and the encryption vector are distinctly separate security measures. The status of one has no effect on the other, and each may be used or not used independently of the other.

Program via Script

Program the device using a pre-defined script (see Script Definition Menu below) to step through a series of operations.

Status

This command causes the PC software to read the programmer and to update the information displayed on the Status line of the menu screen. The Status line contains four parameters: Version, COM, Signature, and Option Byte. The Version indicates the revision level of the system software and the programmer's firmware. It is displayed as PC_VERSION/FIRMWARE_VERSION. The COM parameter indicates the currently selected PC COM port. Signature displays the three signature bytes read from the device during its last access (see data sheet for description). The Option Byte displays the option byte read from the device during its last access (see data sheet for description).

Script Definition Menu

This menu allows the user to define a script that will cause a specific series of operations to be performed when it is executed. The script is initiated by executing a "Program Via Script" command from the PC hosted system software or by pressing the programmer's Program button. Pressing the program button always executes the script currently stored in the programmer's non-volatile memory. The programmer's non-volatile memory is updated whenever the user confirms a modification to the script definition from the menu software.

A script contains individual programmer commands and associated options that are executed in sequence, one at a time. Available script commands are identically the commands available under the device menu (that is, blank check, read, verify, iblank check, program, serialize, option byte, write encryption, and write lock bits). The serialize command has no options when executed from within a script. For this function to work properly here, its options must be defined using the serialize menu of the system software. Once set, the serialize options are also stored in the non-volatile memory of the programmer.
When the Script Definition menu is selected, the user is presented with two main windows. The Option Selection window contains a list of available commands, and the Script Definition window contains the

current script. A blinking cursor shows the currently selected window, and the cursor may be moved from window to window using the right and left arrow and tab keys. Both windows contain a place marker that may be moved up or down by placing the blinking cursor in the desired window, and then using the up and down arrow keys.

Within a script, each command may be included, omitted, or repeated as desired (with certain restrictions). When combined, they simply form a sequence of operations that will be performed when the script is executed. There are certain limitations built into the system that restrict the order and occurrence of some of the instructions. For instance, there must always be at least one program command in a script since the programmer's Program button initiates the execution of the currently defined script. Also there are certain logical restrictions. For instance, you would not program a device and then check to see if it is blank. There are a number of restrictions like this. If any of these are entered into the script, a warning message will be displayed and the script will not be modified. There is a limit of 16 on the total number of commands that can be part of the script. A simple example of a script follows:

Intelligent Blank Check	0 FF
Blank Check	100 3FFF
Program	0 FF
Verify	0 FF
Serialize	
Write Encryption Vector	
Write Lock Bits	Level 4

In this example, the area of memory in which program information will be stored is checked (intelligent blank check) to see that the desired values can be programmed. The remainder of the device is checked to see that it is blank, so that the serialize function will have a clear area in which to write. The program information is stored in the device, it is verified, the device is serialized, the encryption vector is written to the device, and finally the device is locked.

System Hardware

The hardware of the programmer consists of the main programming module and the plug in power supply. As illustrated in Figure 1, the main programming module contains a 40-pin socket for the device to be programmed, and a number of switches and lights for system operation. If the programmer has been initialized with the desired programming script, the programmer may be operated in a stand alone mode.

In Stand alone mode, the programmer's functions are initiated from the front panel. There are four front panel buttons that control operation: Blank check, Program, Verify, and Read. In addition, there are three lights that indicate the programmer's status: Busy, Complete, and Error.

At the top of the zero insertion force socket, recessed into the case, there is a small switch. This switch changes the programmer’s internal controlling processor from operating to loader mode. This switch is provided to allow the user to install future firmware updates. **The switch should never be moved from its leftmost position. Doing so can erase the controller's firmware and make the programmer inoperable.**

Read Button

The Read button causes all of the EPROM in the device to be read, and the contents stored in the programmer's buffer memory. When the read process begins, the Busy light blinks to indicate that the programmer is performing the requested operation. When the read operation is complete and no error has occurred, the Complete light flashes. If an error occurs during the operation, the Error light flashes.

Since the Read operation destroys the existing contents of the programmer’s buffer, the button does not activate the function immediately. The user is required to hold the button down for two seconds before the operation begins. This eliminates the possibility of inadvertently initiating a read.

If none of the EPROM lock bits are set, and the device's encryption vector is blank (that is, all 1’s), then the data output from the device during a read operation is precisely the data stored in its EPROM. However if a value other than all 1’s is stored in the device’s encryption vector, the device will perform encryption on the data as it is read out. This encryption process is the normal security measure offered by the encryption vector. It should be noted that if a read is performed after the encryption vector is written, the programmer’s buffer memory will be overwritten with encrypted data. This is typically not the desired result.

Lock bits also affect the read operation. If either of the upper two levels of lock (level 3 or 4) are set, the device will output all 1’s when it is read. Therefore, the only way to distinguish between a blank device and one that is locked is to try to program it. If a single location is programmed, and the program operation fails, then the device is potentially locked.

Verify Button

When the Verify button is pressed, the Busy light begins flashing and the entire contents of the device's EPROM memory are compared with the contents of the programmer's buffer memory. If the two contain the same data, the Complete light flashes, but if there is a difference, the Error light flashes.

Since the verify operation performs a read of the device, the same comments on the encryption vector and lock bits made in the Read Button section above also apply to the verify operation.

Blank Button

The blank button confirms that all of the device's EPROM memory is blank (all 1’s). The Busy light blinks to indicate that the function is being performed and no error has been detected. If an error is detected (the EPROM is not all 1’s), the process stops immediately and the Error light begins flashing. If the device is blank, the Complete light will flash.

Note that the comments regarding the encryption vector and lock bits made in the above sections apply to the Blank button as well.

Program Button

The Program button initiates a sequence of events that are specified by the current script (see “Script Definition” section above). One of these events will be to transfer data from programmer's buffer memory into the device's EPROM (that is, program the device). While these programming events are being performed, the Busy light blinks. If an error occurs, the process is stopped immediately, and the Error light begins flashing. If all operations are successful, the Complete light will flash.

By allowing the program button to initiate a script, complex operations may be performed in the absence of additional equipment. This allows the programmer to be configured where a PC is conveniently available, then easily carried to where the devices are actually programmed (possibly an assembly area) without the need for additional equipment. Note that since the script settings are stored in non-volatile memory, the user should verify them before a programming session to ensure the device(s) will be programmed as desired.

Like the Read button, the Program button must be pressed for 2 seconds before the programming function begins. This feature protects against accidental initiation of the program operation.

Installation

System Software

The system software for the DS87000 is provided on a floppy disk. Copy the contents of the disk to the desired hard disk directory or a working floppy disk and store the distribution disk safely for future use.

At the DOS prompt, enter the command "DS87000". This will start the menu driven system software. Using the mouse or "hot keys" (shown on the menu), enter the DOS sub-menu and select the COM port to which the programmer is connected. Communications with the programmer will begin automatically. The program can also be started with the port number as a command line option. For instance, if the programmer is attached to port 2, the command “DS87000 2” can be entered at the DOS prompt.

Hardware

Installing the DS87000 hardware consists of plugging the power supply into a wall outlet and connecting the supplied RS-232 cable. The United States version of the programming kit (DS87000-000) includes a standard 110 volt power transformer. A 220-volt transformer is provided with the DS87000-220 version of the programmer. This transformer is equipped with a two prong cylindrical plug, and may require a mechanical adapter for operation in some areas. The end of the transformer's cable contains a D.C. power connector that plugs into the opening in the rear wall of the programmer. When power is applied, the red LED will illuminate.

The programmer kit also contains an RS-232 cable with RJ11 connectors on both ends. Please note that this is not a standard telephone cable, and **replacing it with one may cause damage**. Plug one end of the cable into the mating connector on the top of the programmer. The other end of the cable should be plugged into the supplied RJ-11-to-DB9 adapter. That adapter is then connected to an available serial COM port on the PC. The DS87000 system software allows the use of COM ports 1, 2, 3, or 4.

Note:

Ports 1 and 3 usually share an interrupt as do 2 and 4. The DS87000 assumes this is the configuration of the PC on which it is running. Because of this interrupt arrangement, the user must ensure that there is no conflict with other equipment.

System Specifications

Parameter	Specification
Storage Temperature	-40 °C to +70 °C
Operating Temperature	0 °C to 70 °C
AC Input (power supply)	105-129 V @ 60 Hz (DS87000-000) o 198-242 V @ 50 Hz (DS87000-220)
DC Input (programmer)	16.0 V @ 100 mA (MIN), 19.0 V @ 200 mA (MAX)
Serial Communications	RS-232