App Note 3398: DS80C400/DS80C410/DS80C411 Network Boot

This application note describes "Network Boot," or (NetBoot) and gives the hardware requirements. A feature built into the ROM of Dallas Semiconductor's networked microcontrollers, NetBoot is a fast, easy way to program flash and nonvolatile SRAM memories in a production environment. NetBoot loads user application code and user application data into SRAM or AMD® AM29LV081B-compatible flash memory, after retrieving it from the network using TFTP.

Overview

Network Boot (NetBoot) is a feature built into the ROM of Dallas Semiconductor's networked microcontrollers (DS80C400, DS80C410 and DS80C411). The NetBoot feature allows fast and easy programming of flash and nonvolatile SRAM memories in a production environment.

NetBoot loads user application code and user application data into SRAM or AMD® AM29LV081B-compatible flash memory. Code and data are retrieved over the network using the industry standard TFTP protocol.

This application note describes the hardware requirements for NetBoot and explains how NetBoot works.

All program code referenced in this application note is available for download from the Dallas Semiconductor ftp server at ftp://ftp.dalsemi.com/pub/tini/ds80c400/netboot/.

Requirements

NetBoot requires the following:

- A networked microcontroller (DS80C400, DS80C410 or DS80C411)
- A PHY interface chip
- A DS2502-E48 to store the MAC ID (also called "Ethernet address")
- At least 48kB of SRAM
- A crystal speed of 27MHz or higher for 100MB operation
- A TFTP server on the network

PHY

The PHY has to be configured at PHY address 0, and multiple PHYs are not supported by NetBoot. Furthermore, on the DS80C400, RSTOL must not be connected to the PHY's reset input. (See DS80C400 datasheet. This restriction does not apply to the DS80C410/DS80C411.)

MAC ID

The preprogrammed version of the DS2502, the DS2502-E48, already contains a globally-unique Ethernet MAC ID. Using the DS2502-E48 is the most convenient and safest way to assign a MAC address. The DisplayMAC2502 Java® program (see ftp server) displays the contents of the DS2502-E48 on the TINIm400 evaluation module.

Starting with the DS80C410/DS80C411 (NetBoot 1.2.0 and later), customers who prefer to assign their own MAC ID ranges can also program generic DS2502s. The required data format is listed in the DS2502-E48 data sheet. The

example program (Listing 1) shows how to program a DS2502 (also available in an iButton package as DS1982). Note that a 12V programming pulse is required, which is not supported by the 1-Wire master on the DS80C410/DS80C411. Instead, a PC and the DS9097U-E25 1-Wire adapter are used for programming.

Listing 1-Programming a MAC address into a DS2502 (WriteMAC2502.java)

```java
/*
 * WriteMAC2502.java: Write MAC ID to DS2502/DS1982.
 * To be run from a PC with a DS9097U-E25 adapter.
 *
 * To build this software, run the following command:
 * javac -classpath OneWireAPI.jar WriteMAC2502.java
 *
 *-------------------------------------------------------------------
 * Copyright (C) 2004 Dallas Semiconductor Corporation, All Rights Reserved.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
 * OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
 * ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
 * OTHER DEALINGS IN THE SOFTWARE.
 *
 * Except as contained in this notice, the name of Dallas Semiconductor
 * shall not be used except as stated in the Dallas Semiconductor
 * Branding Policy.
 *-------------------------------------------------------------------
 */

import java.util.*;
import com.dalsemi.onewire.*;
import com.dalsemi.onewire.adapter.*;
import com.dalsemi.onewire.container.*;
import com.dalsemi.onewire.utils.CRC16;

public class WriteMAC2502
{

  public static boolean program2502(OneWireContainer owc, byte[] macid)
   {
     byte[] pageData = new byte[32];
     int i;

     try {
```

```java
      Enumeration enumbank = owc.getMemoryBanks();
      PagedMemoryBank pmb = null;
      while (enumbank.hasMoreElements()) {
        MemoryBank mb = (MemoryBank)enumbank.nextElement();
        if (mb.isWriteOnce()) {
          pmb = (PagedMemoryBank)mb;
          break;
        }
      }

      if (pmb == null) {
        System.out.println("STOP: Could not find a paged memory bank");
        return false;
      }
      else {
        pmb.readPage(0, false, pageData, 0);
        for (i = 0; i < 32; i++) {
          if ((pageData[i] & 0xff) != 0xff) {
            System.out.println("STOP: DS2502/DS1982 already programmed");
            return false;
          }
        }

        // Contents of a DS2502-E48
        pageData[0] = (byte) 0x29; // Project ID
        pageData[1] = (byte) 0x11;
        pageData[2] = (byte) 0;
        pageData[3] = (byte) 0;

        // Reverse copy the MAC ID we want to write
        for (i = 5; i >= 0; i--)
          pageData[9-i] = macid[i];

        System.out.println();
        System.out.print("Writing...");

        pmb.writePagePacket(0, pageData, 0, 10);

        byte[] verify = new byte[32];
        pmb.readPage(0, false, verify, 0);

        System.out.println();
        System.out.print("New Contents of Page 0:");
        for (i = 0; i < 32; i++) {
          if (i % 16 == 0)
            System.out.println();
          if ((verify[i] & 0xff) < 0x10)
            System.out.print("0");
          System.out.print(Integer.toHexString(verify[i] & 0xff) + " ");
        }
        System.out.println();
        System.out.println();

        for (i = 0; i < 10; i++) {
          if (pageData[i] != verify[i+1]) {
            System.out.println("STOP: Programming failed (offset " + i + ")");
```

```java
          return false;
        }
      }
    }
    System.out.println("Done.");
    return true;
  }
  catch (Exception e) {
    System.out.println("Exception " + e.toString());
    return false;
  }
}

public static void main(String args[])
{
  DSPortAdapter adapter;
  byte[] macid = new byte[6];

  // Set MAC ID here
  // 00:01:02:03:04:05 is NOT a good idea!
  // Register your MAC ID range with the IEEE.
  macid[0] = (byte) 0x00;
  macid[1] = (byte) 0x01;
  macid[2] = (byte) 0x02;
  macid[3] = (byte) 0x03;
  macid[4] = (byte) 0x04;
  macid[5] = (byte) 0x05;

  try {
    adapter = OneWireAccessProvider.getDefaultAdapter();
    adapter.beginExclusive(true);
    if (!adapter.canProgram()) {
      System.out.println("STOP: Adapter does not support 12V programming.");
      System.out.println("      DS9097U-E25 required, or 12V supply not present.");
      return;
    }
  }
  catch (Exception e) {
    System.out.println("STOP: No adapter / " + e.toString());
    return;
  }

  OneWireContainer owc;
  byte[] targets = { 0x09 };  // DS2502/DS1982
  adapter.targetFamily(targets);

  try {
    owc = adapter.getFirstDeviceContainer();

    if (owc != null) {
      System.out.println("Found DS2502/DS1982: " + owc.getAddressAsString());
      program2502(owc, macid);
    }
    else
      System.out.println("STOP: No DS2502/DS1982 found.");
```

```
      }
    catch (Exception e) {
      System.out.println("Exception " + e.toString());
      }
    }
  }
}
```

Very important: When assigning a custom MAC ID, it is important to only used IDs registered with the IEEE®. Under NO circumstances select a random MAC address or the address of another existing device. MAC addresses are globally unique and network stability depends on well-behaved devices!

Remember that the IEEE registration and 1-Wire programming steps are not supported on the DS80C400 (NetBoot 1.0.1). Registration and programming are NOT required with a DS2502-E48-it works straight out of the box.

SRAM
NetBoot on the DS80C400 requires at least 48kB of SRAM connected to CE0. Both the DS80C410 and DS80C411 have an additional 64kB block of internal SRAM, and no external SRAM is required for NetBoot on these processors.

Oscillator Frequency
NetBoot does not use the crystal multiplier. The crystal speed is, therefore, the operating speed of NetBoot. To operate on 100Mbit networks, the crystal speed needs to be at least 27MHz. Otherwise, the PHY needs to be configured to allow only 10Mbit operation.

Networking Environment
The network should be low in multicast or broadcast messages and there must be a fully functional TFTP server on the network. Unfortunately, installation and configuration of a TFTP server are beyond the scope of this article. Windows® users might want to try the SolarWinds® TFTP server; Linux® and Macintosh® users can use the built-in tftpd.

Starting Netboot
There are four ways to start NetBoot:
'N' Loader Command
NetBoot can be invoked from the ROM loader using the 'N' command. When using the 'N' loader command, NetBoot will print messages to serial port 0-a very useful debugging aid. (See listing 2)

Listing 2 - Example NetBoot output

```
DS80C400/DS80C41X Silicon Software - Copyright (C) 2002-2004 Maxim Integrated Products
S/N: 4F5E7000C246C689 MAC ID: 00603515DEAD
Starting DHCP... IP leased.
No TFTP server found.
```

Port Pin/Reset
The second method of starting NetBoot is to pull port pin P5.3 low and reset the CPU. On the TINIm400 evaluation module, place jumper J3 to set P5.3 and then reset the CPU.

System Call
ROM NetBoot can also be started by application software. (The TINI® OS method is com.dalsemi.tininet.NetBoot. The C library function is called init_netboot) This approach is useful when the user application software needs to retain some control over when to invoke NetBoot, while the ROM only tests the state of one port pin, user application software can evaluate more complex, customer-defined decision logic.

Library Function
In addition to NetBoot built into ROM, Dallas Semiconductor offers an IPv4-only C library (xnetboot) that is more configurable than the in-ROM versions of NetBoot. However, the library cannot be used to load code into the same memory from which the library code is running. For example, when NetBoot is located in flash memory connected to CE3, it can be used to load another flash memory connected to CE2.

The C library function

- Sets the clock multiplier to 1, 2, or 4 (allowing a more flexible choice of crystals)
- Disables multicast reception for improved performance on IPv4 networks.
- Offers the latest version of NetBoot for all processors.

Netboot Process
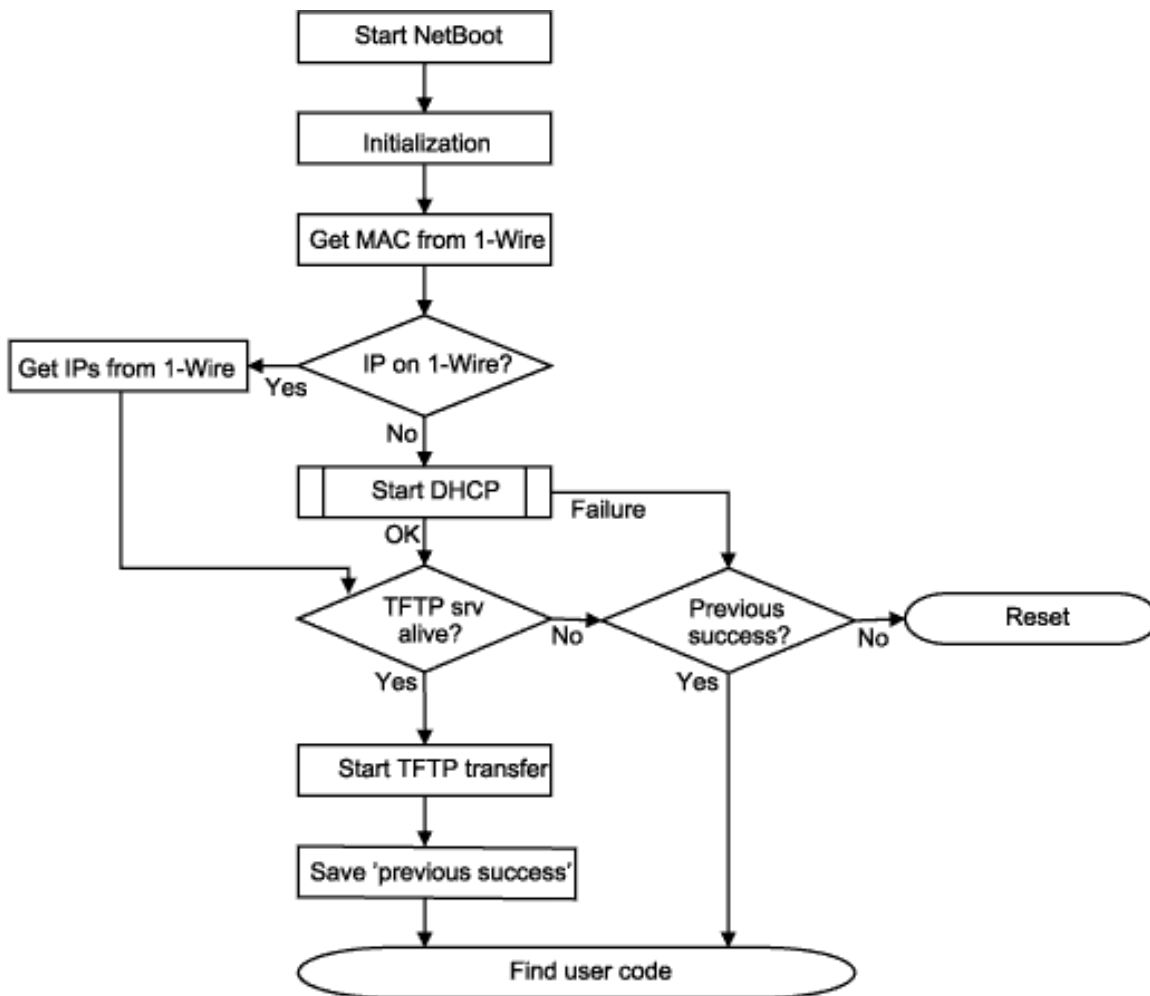Figure 1 shows a flow chart of the NetBoot process.

Figure 1 - NetBoot

The following paragraphs explain the different steps in more detail.

Memory Initialization
On the DS80C400, NetBoot clears the first 64kB of SRAM connected to CE0. On the DS80C410/DS80C411, NetBoot clears the internal 64kB SRAM.

MAC Address Acquisition
The MAC address is read from the DS2502(-E48) connected to the 1-Wire bus. When a DS2502(-E48) cannot be found, NetBoot will fail.

IP/TFTP Server Configuration

Next, 1-Wire is also checked for the presence of an optional user-programmed memory device (as shown in Table 1), for example a DS2433. If a memory device is found, it is examined for static IP address configuration, and the IP address of the TFTP server.

Table 2 shows the data format of the optional memory device.

Table 1. Configuration Device Family Code Reference

Family Code	Part Number (iButton Package)	Description (memory size in bits)
04h	DS1994 (DS2404)	4k NV RAM memory and clock, timer, alarms
06h	DS1993	4k NV RAM memory
08h	DS1992	1k NV RAM memory
0Ah	DS1995	16k NV RAM memory
0Ch	DS1996, DS1996x2, DS1996x4	64k to 256k NV RAM memory
14h	DS1971 (DS2430A)	256-bit EEPROM memory and 64-bit OTP register
23h	DS1973 (DS2433)	4k EEPROM memory

Table 2. Configuration Device Memory Contents

Offset	Length	Description
0	1	Size of configuration information (must be 29)
1	4	Configuration tag (must be 'TINI')
5	4	Static IPv4 address (e.g. 192.168.0.10), or 0
9	4	Static IPv4 gateway (e.g.192.168.0.1), or 0
13	1	Bit length of IPv4 netmask (e.g. 24 for 255.255.255.0), or 0
14	16	IPv6 address of TFTP server-OR- 12 zero bytes followed by IPv4 address of TFTP server
30	2	1-complement CRC-16 of offset 0-29

The following example program (listing 3) shows how to program a DS2433.

Listing 3 - Programming a DS2433 as NetBoot Configuration Memory (WriteIP2433.java)

```
/*
 * WriteIP2433.java: Write NetBoot IP configuration to DS2433.
 *
 * To build this software, run the following command:
 * javac -classpath OneWireAPI.jar WriteIP2433.java
 *
 *-------------------------------------------------------------------------
 * Copyright (C) 2004 Dallas Semiconductor Corporation, All Rights Reserved.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
```

```
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY,  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL DALLAS SEMICONDUCTOR BE LIABLE FOR ANY CLAIM, DAMAGES
 * OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
 * ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
 * OTHER DEALINGS IN THE SOFTWARE.
 *
 * Except as contained in this notice, the name of Dallas Semiconductor
 * shall not be used except as stated in the Dallas Semiconductor
 * Branding Policy.
 *---------------------------------------------------------------------------
 */

import com.dalsemi.onewire.*;
import com.dalsemi.onewire.adapter.*;
import com.dalsemi.onewire.container.*;
import com.dalsemi.onewire.utils.CRC16;

public class WriteIP2433
{
  static final int TARGET_FAMILY_ID        = 0x08; // DS2433
  static final int READ_MEMORY_COMMAND     = 0xf0;
  static final int WRITE_SCRATCHPAD_COMMAND = 0x0f;
  static final int COPY_SCRATCHPAD_COMMAND  = 0x55;
  static final int READ_SCRATCHPAD_COMMAND  = 0xaa;

  public static void main(String[] args)
  {
    DSPortAdapter adapter;
    try {
      adapter = (DSPortAdapter)(new TINIInternalAdapter());
    } catch (Exception e) {
      System.out.println("No adapter / " + e.toString());
      return;
    }
    boolean foundIt = false;

    try {
      long deviceAddress;

      adapter.beginExclusive(true);
      if (adapter.findFirstDevice()) {
        // Test LSB (family id) against target
        deviceAddress = adapter.getAddressAsLong();
        if ((deviceAddress & 0xff) == TARGET_FAMILY_ID)
          foundIt = true;
        while (!foundIt && adapter.findNextDevice()) {
          System.out.println("Family: " + (deviceAddress & 0xff));
          deviceAddress = adapter.getAddressAsLong();
          if ((deviceAddress & 0xff) == TARGET_FAMILY_ID)
```

```java
                foundIt = true;
            }
            if (foundIt) {
                byte[] command = new byte[4];
                command[0] = (byte) READ_MEMORY_COMMAND;
                command[1] = 0;
                command[2] = 0;
                adapter.select(deviceAddress);
                adapter.dataBlock(command, 0, 3);

                byte[] IPdata = adapter.getBlock(32);

                for (int i = 0; i < 32; i++)
                    System.out.print(Integer.toHexString(IPdata[i] & 0xff));
                System.out.println();

                int crc = CRC16.compute(IPdata);
                System.out.println("Original CRC=" + Integer.toHexString(crc));

                byte[] config = new byte[32];
                config[0]  = 29;           // Length
                config[1]  = 'T';
                config[2]  = 'I';
                config[3]  = 'N';
                config[4]  = 'I';
                config[5]  = (byte) 192; // Static IPv4 address
                config[6]  = (byte) 168;
                config[7]  = (byte) 0;
                config[8]  = (byte) 10;
                config[9]  = (byte) 192; // Static IPv4 gateway
                config[10] = (byte) 168;
                config[11] = (byte) 0;
                config[12] = (byte) 1;
                config[13] = (byte) 24; // Net mask (255.255.255.0)
                config[14] = 0; // TFTP server IP
                config[15] = 0;
                config[16] = 0;
                config[17] = 0;
                config[18] = 0;
                config[19] = 0;
                config[20] = 0;
                config[21] = 0;
                config[22] = 0;
                config[23] = 0;
                config[24] = 0;
                config[25] = 0;
                config[26] = (byte) 192; // IPv4 part of TFTP server IP
                config[27] = (byte) 168;
                config[28] = (byte) 0;
                config[29] = (byte) 17;

                crc = ~CRC16.compute(config, 0, 30);
                config[30] = (byte) (crc & 0xff);
                config[31] = (byte) ((crc >> 8) & 0xff);
```

```java
        for (int i = 0; i < 32; i++)
           System.out.print(Integer.toHexString(config[i] & 0xff));
        System.out.println();

        command[0] = (byte) WRITE_SCRATCHPAD_COMMAND;
        command[1] = 0;
        command[2] = 0;
        adapter.select(deviceAddress);
        adapter.dataBlock(command, 0, 3);
        adapter.dataBlock(config, 0, 32);

        command[0] = (byte) READ_SCRATCHPAD_COMMAND;
        adapter.select(deviceAddress);
        adapter.dataBlock(command, 0, 1);
        byte[] scratch = adapter.getBlock(3+32);

        for (int i = 0; i < 32+3; i++)
           System.out.print(Integer.toHexString(scratch[i] & 0xff));
        System.out.println();

        command[0] = (byte) COPY_SCRATCHPAD_COMMAND;
        command[1] = 0;
        command[2] = 0;
        command[3] = scratch[2];
        adapter.select(deviceAddress);
        adapter.dataBlock(command, 0, 4);
      }
      else
        System.out.println("Device not found");
    }
  }
  catch (OneWireException owe) {
    System.out.println(owe.getMessage());
  }
  finally {
    adapter.endExclusive();
  }
 }
}
```

If there is an IPv6 TFTP server address configured, the following test is skipped.

The TFTP server address is determined (in order of decreasing priority)

1. From the 1-Wire memory device, if present (IPv6 or IPv4)
2. From the DHCP user option 150, if present (IPv4 only)
3. From the DHCP "next server" field, if present (IPv4 only)

NetBoot will fail when a TFTP server IP cannot be determined. Otherwise, if the static IPv4 address fields are 0, a DHCP client is started to acquire an IPv4 address.

TFTP Files
Once all IP addresses are configured, NetBoot tries to contact the TFTP server and starts asking for several different files (which may or may not be present on the TFTP server). When a TFTP request for a specific file is successful,

the file is fetched from the TFTP server and programmed into memory. The loader automatically determines whether a memory location is SRAM or flash, and sector erases flash memory when necessary.

Table 3 lists the file names and shows the order in which these files are requested.

Table 3. Sequence of File Names Requested by NetBoot

Seq. #	File Name	Example File Name	Notes
1	'!'MACID	!006035010203	Only NetBoot version 1.2.0 and later. When this request is successful, NetBoot is immediately terminated (no file is loaded at all).
2	MACID	006035010203	Unique for each device
3	'TINI400-'version	TINI400-1.0.1 (DS80C400 B1) TINI400-1.2.0 (DS80C410 A1)	Unique for each ROM revision
4	'TINI400'	TINI400	All compatible Dallas CPUs

The "Previous Success" Indicator
When TFTP succeeds, it writes the signature 'TIN' to memory location 000100h. Conversely, this memory location is cleared on failure. This process could be exploited by a user application to better control NetBoot. However, both TINI OS and the C libraries ignore/clear this memory location.

Search for Code
Once NetBoot terminates, the ROM determines which user application code to run. Memory is searched downwards in 64kB steps for a signature. As soon as a valid signature is detected, control is transferred to the application code. Note that a user application at a higher memory address will always win over an application at a lower address.

On the DS80C410/DS80C411, there is a special exception to this rule: Memory location C000h in the internal SRAM (the last 16kB of the internal SRAM) is examined first.

Table 4 shows the format of the tag required for execution.

Table 4. Format of the Code Tag

Offset	Length	Description
0	2	SJMP instruction to application code
2	4	Tag—must be 'TINI'
6	1	Segment—0, or must match the high 8 bits of the tag's memory address

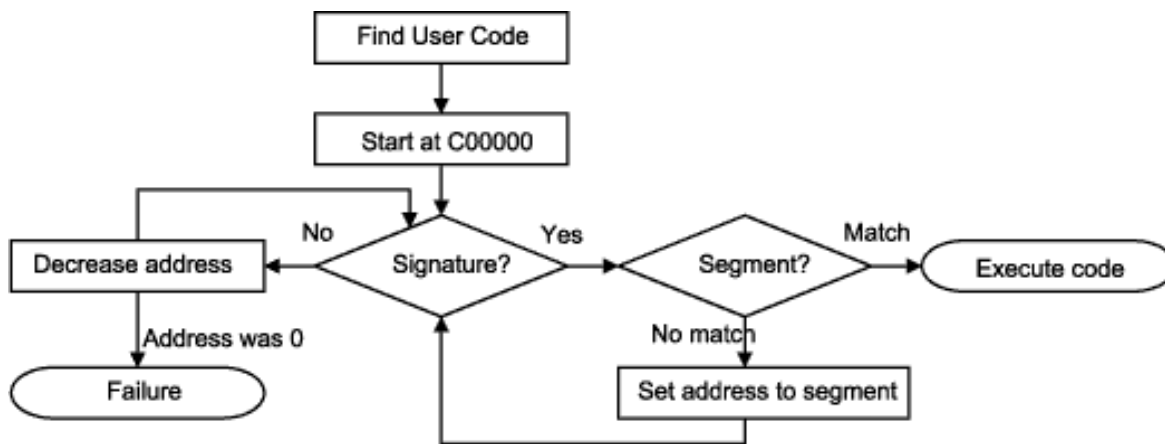Figure 2 illustrates how NetBoot locates user application code in external memory.

Figure 2 Determining the Start Address of Code

Format of Netboot Files

The NetBoot file format is called the 'tbin2' format. Table 5 shows the definition of a 'tbin2' format record. A tbin2 file can contain any number of records, however, as the records are programmed into memory sequentially, overlapping memory ranges should be avoided. Also, there should be only one record per flash sector.(Otherwise, a flash sector erase could destroy data written previously.)

Table 5. tbin2 Format Record

Offset	Length	Description
0	1	Version of tbin2 record—must be 1
1	3	Start address of data (LSB first)
4	2	Length-1 of data (LSB first)
6	*	Data

Note that NetBoot version 1.0.1 only supports files smaller than 64kB. However, there is a workaround implemented in two converters available from the ftp site. (The workaround reorders records and changes CRC values.)

Table 6 lists the tbin2 converters available from the Dallas Semiconductor ftp site.

Table 6. Conversion Programs Available from the ftp Site

Name	Description
hex2tbin2.c	Converts an Intel HEX-386 file to tbin2 (HEX-386 is generated by the Keil C compiler)
tbin2tbin2.c	Converts a TBIN file into tbin2 (TBIN is used by the TINI OS)
hex2fw.c	A version of hex2tbin2.c that works around the 64 KB limitation in NetBoot 1.0.1
tbin2fw.c	A version of tbin2tbin2.c that works around the 64 KB limitation in NetBoot 1.0.1

Conclusion

NetBoot replaces the serial loader, allowing a streamlined production process for all users of the Dallas Semiconductor networked microcontrollers.

References

All program code referenced in this application note is available for download from the Dallas Semiconductor ftp server at ftp://ftp.dalsemi.com/pub/tini/ds80c400/netboot/.

Please refer to the High-Speed Microcontroller User's Guide: Network Microcontroller Supplement and the DS80C400/DS80C410/DS80C411 data sheets for more details about the networked microcontrollers.

The C library web site for the DS80C400, DS80C410 and DS80C411 A user discussion board is available.

AMD is a trademark of Advanced Micro Devices, Inc.
TINI and 1-Wire are registered trademarks of Dallas Semiconductor.
Java is a trademark of Sun Microsystems.
Windows is a registered trademark of Microsoft Corp.
SolarWinds is a registered trademarks of SolarWinds.net, Inc
Linux is a registered trademark of Linus Torvalds.
Macintosh is a registered trademark of Apple Computer, Inc.
IEEE is a registered trademark of the Institute of Electrical and Electronics Engineers.

More Information

DS80C400: QuickView -- Full (PDF) Data Sheet -- Free Samples

DS80C410: QuickView -- Full (PDF) Data Sheet -- Free Samples

DS80C411: QuickView -- Full (PDF) Data Sheet -- Free Samples