1. **Overview**

1.1 **Scope**

This document describes how to use .NET Micro Framework (NETMF) with SH7619 EVB board. This document mainly describes how to setup, build and run SH7619_EVB solution available in the porting kit. This document also describes how to create, build, deploy and run NETMF based applications on SH7619 EVB board using Microsoft visual studio.
1.2 Required Target System

- SH7619 EVB

Please go through the document of the SH7619 EVB before power ON and working with the board.
1.3 **Supported Driver**

- Serial
- On chip Ethernet
- Timer & Power
- Nor Flash

Following drivers are just sample. In the SH7619 EVB there is no LCD panel and no Key Input.
If you do some wire arrangement, following sample drivers might help you to create driver.
- Display (LCD: 132 x 176 TFT LCD Display (HD66773R) with 262,144 colors)
- Key input using GPIO.
1.4 Supported Projects

Following projects are supported.
- NativeSample
- TinyCLR

Following projects are not supported.
- Portbooter
- Tinybooter

However, these projects can be build properly so if needed you can customize them.
1.5 Requirements

- Host system (Windows XP in this documentation)
- Target System (SH7619 EVB)
- Microsoft Visual Studio 2010
- E10A-USB Emulator for SH2 and installation setup.
- C/C++ Compiler Package for SuperH RISC engine family
- .NET Micro framework porting kit (MicroFrameworkPK.msi)
- .NET Micro framework development kit (MicroFrameworkSDK.MSI)
- .NETMF Solution “SH7619_EVB” (included in the porting kit)
- NULL modem cable (Serial cable)
- Tera Term Pro or another serial program.
2. **Setup**

2.1 **Preparation**

Before setting up Porting Kit, below software have to be installed.

- Visual Studio 2010
- Renesas C/C++ Compiler Package for SuperH RISC engine family V.9.02 Release 00 or later

For these instructions, we will assume it is installed in

```
c:\Program Files\Renesas\Hew\Tools\Renesas\Sh\x_y_z*
```
2.2 Install .NET Micro Framework Porting Kit

Install porting Kit by executing MicroFrameworkPK.msi.
2.3 Set Path

1) Open a command-line prompt
2) Change the current directory to
   C:\MicroFrameworkPK_v4_1
3) Enter below,
   setenv_shc "c:\Program Files\Renesas\Hew\Tools\Renesas\Sh\x_y_z*"

*x_y_z is the version of the compiler being used for example 9_2_0.
3. Memory Map

The memory map of SH7619 EVB .NETMF Porting kit is shown below.

![Memory Map Diagram]

**Figure 3.1** The memory map of SH7619 EVB .NETMF Porting Kit
4. **How to build and execute**

In this section, we shall describe the way to build, download and execute SH7619 EVB solution available in the porting kit with NOR Flash Memory and SDRAM.
4.1 How to Build

1) Using Command-line prompt, change the directory to “Solutions\SH7619_EVB”

    C:\MicroFrameworkPK_v4_1>cd solutions\SH7619_EVB

2) Run

    Msbuild dotnetmf.proj /t:build
    /p:flavor=debug;EnableTcpIp=true;TCP_IP_STACK=LWIP

    Flavor: <debug|release|rtm>
    EnableTCPIP : <true|false>
    TCP_IP_STACK : LWIP     If you want to use RTIP, don’t need to add
    TCP_IP_STACK.

    If you want to debug your program using E10A-USB Emulator, please
specify “debug” for “flavor” option.
4.2 Board switch settings

Set SW1 of the SH7619 EVB board as below to startup from NOR Flash.

<table>
<thead>
<tr>
<th>SW</th>
<th>Setting</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ON (Low)</td>
<td>MD0</td>
</tr>
<tr>
<td>2</td>
<td>OFF (High)</td>
<td>MD1</td>
</tr>
<tr>
<td>3</td>
<td>ON (Low)</td>
<td>MD2</td>
</tr>
<tr>
<td>4</td>
<td>ON (Low)</td>
<td>MD3</td>
</tr>
<tr>
<td>5</td>
<td>ON (Low)</td>
<td>MD5</td>
</tr>
</tbody>
</table>

The meaning of above setting is below,

- Clock mode
  Clock Mode is MODE2.
  Each frequency is below,
  - Crystal 15.36MHz
  - CPU 15.36MHz
    FRQCR register is set by software as 0x1103 so CPU frequency will be 122.88MHz.
  - BUS 61.44MHz
  - Peripheral 30.72MHz

- CS0 Memory bus width
  16 Bit

- Endian
  Big endian
4.3 Download using E10A-USB

1) Setup the E10A-USB Emulator

   Install the E10A-USB Emulator software into your PC.
   During install, you should select the device group for E10A-USB then specify “Super H RISC engine family SH-2 device group”

2) How to start downloading using E10A-USB

   Extract the HEW workspace for SH7619 EVB Porting Kit from file SH7619board.zip at any place in your PC.
   In the following instructions, we will assume it is installed in “c:\workspace”

   Start the Hew with choosing below menu

   ![Figure 4.1 Hew menu](image)

   Then Hew will be show up and you can see below dialog box.

   Specify the workspace as below, and press OK button.

   ![Figure 4.2 Welcome dialog box](image)
In below dialog box, choose “SH7619”.

![CPU Select dialog box](image)

**Figure 4.3  CPU Select dialog box**

When below message shows up, power ON the target board and then press OK button.

![heu7618 dialog box](image)

**Figure 4.4  heu7618 dialog box**

Then Hew/E10A-USB will finish connection with SH7619.
Figure 4.5  Hew window
3) How to download the program to the flash memory

i) Prepare the download module.

Select the [Debug] -> [Debug Settings...] from the menu bar of Hew then below dialog box will show up.

![Debug Settings dialog box](image)

Figure 4.6  Debug Settings dialog box

Here is the definition of download modules. Please change the Path setting for each download module by clicking “Modify” button.
ii) Prepare for downloading to Flash Memory.

Select the [Setup] -> [Emulator] -> [System...] from the menu bar of Hew then “Configuration” dialog box will show up.

Select “Loading flash memory” tab. Then, specify as same as below figure.

For “File name”, please specify

<Workspace folder>\SH7619board\Tools\4MB\fmtool.mot.

![Configuration dialog box](image)

**Figure 4.7  Configuration dialog box**
ii) CPU and memory initialization

In order to download data to flash properly, CPU and memory should be initialized as following.

- CPU: Cache must be disabled.
- Memory: SDRAM must be initialized.

In order to initialize them, run the batch file as below,

- Specify the batch file

![Set Batch File](image)
On the command line window, do the right clicking so that the popup menu shows up.

Select “Set Batch file…” and specify the batch file name like below,

-Run the batch file

After specify the batch file, please press “Play” button so that batch file starts to run.
iii) Download module to Flash

In order to download, double click on the name of the download module which you want to download to flash. It takes more than 30 seconds.

In the case of downloading TinyCLR, please select tinyclr.abs instead of nativesample.abs.
5. **Running TinyCLR**

Hit “Stop” button in the toolbar or Select “Halt Program” option from Debug menu. Disconnect E10A-USB from HEW, power OFF the board and disconnect E10A connection from board.

To verify that TinyCLR is up and running connect the serial port of the target with PC and start the terminal program Tera Term Pro with following settings:

![Serial port setup](image)

**Figure 5.1 Serial port setup**

Power ON the board and you should see following messages in the serial terminal:
If you can see above like above, congratulations! Your TinyCLR is up and running. This example is the case of using TinyCLR.abs by debug build.

Please note that don’t forget to close Tera Term Pro before proceeding forward.
6. **How to set the MAC address**

The initial value of MAC address is FF.FF.FF.FF.FF.FF.

There are two ways for changing MAC address.
6.1 Using Serial Terminal

Using Serial Terminal, you can change the MAC address without modifying any source code.

1) Open the Serial terminal and connect the serial port of the SH7619 EVB with PC via Serial cross cable.

2) Press Enter key

3) Power on the SH7619 EVB with pressing Enter key.

Please don’t release the Enter key until you can find following message in terminal window.

4) Enter new MAC address.

   If MAC address is expected 1a.e.f.8.6.f3, type below,

   1a.0e.0f.8.06.f3
6.2 Modifying source file

You can change the MAC address modifying source code.

1) Change the sequence for getting MAC address.

Open C:\MicroFrameworkPK_v4_1\DeviceCode\Targets\Native\sh2\DeviceCode\sh7619\Ethernet\SH7619_EDMAC.cpp

So that you can find RTP_BOOL SH7619_EDMAC_open(PIFACE $Pi$).

Please remove all of following codes.

```c
int addr = MAC_address_area;
for(i=0; i<macLen; i++)
g_NetworkConfig.NetworkInterfaces[NETWORK_INTERFACE_INDEX_SH7619EMAC].macAddressBuf
 = *(volatile char *)(addr+i);

debug_printf( "MAC Address: %x.%x.%x.%x.%x.%x¥r¥n",
  (UINT8)g_NetworkConfig.NetworkInterfaces[NETWORK_INTERFACE_INDEX_SH7619EMAC].macAddressBuf[0],
  (UINT8)g_NetworkConfig.NetworkInterfaces[NETWORK_INTERFACE_INDEX_SH7619EMAC].macAddressBuf[1],
  (UINT8)g_NetworkConfig.NetworkInterfaces[NETWORK_INTERFACE_INDEX_SH7619EMAC].macAddressBuf[2],
  (UINT8)g_NetworkConfig.NetworkInterfaces[NETWORK_INTERFACE_INDEX_SH7619EMAC].macAddressBuf[3],
  (UINT8)g_NetworkConfig.NetworkInterfaces[NETWORK_INTERFACE_INDEX_SH7619EMAC].macAddressBuf[4],
  (UINT8)g_NetworkConfig.NetworkInterfaces[NETWORK_INTERFACE_INDEX_SH7619EMAC].macAddressBuf[5]);

dump_printf( "Do you need to change MAC Address? If yes, press 'Enter' key¥r¥n" );
c[0] = 0x0;
for (i=0; i<0xff; i++){
  DebuggerPort_Read( HalSystemConfig.DebugTextPort, c, 1);
  if (c[0] == 0x0d){
    DebuggerPort_Read( HalSystemConfig.DebugTextPort, c, 1);
    if (c[0] == 0x0d){
      DebuggerPort_Read( HalSystemConfig.DebugTextPort, c, 1);
      if (c[0] == 0x0d){
        DebuggerPort_Read( HalSystemConfig.DebugTextPort, c, 1);
        if (c[0] == 0x0d){
          DebuggerPort_Read( HalSystemConfig.DebugTextPort, c, 1);
          if (c[0] == 0x0d){
            DebuggerPort_Read( HalSystemConfig.DebugTextPort, c, 1);
            Buffer[i] = c[0];
            i++;
          }
        }
      }
    }
  }
}
for (i=0; i<17; i++) hal_printf(个百分.Buffer[i] );
hal_printf( "¥r¥n" );
for(i=0; i<macLen; i++) {
```
Buff[i] = SH7619_EDMAC_AtoH(Buffer[i*3])*0x10;
Buff[i] += SH7619_EDMAC_AtoH(Buffer[i*3+1]);
}

for(i=0; i<macLen; i++) {
    g_NetworkConfig.NetworkInterfaces[NETWORK_INTERFACE_INDEX_SH7619EMAC].macAd
}

download_printf( "Updating...\r\n" );
g_AM29DL_16_BS_DeviceTable.InitializeDevice(pBLOCK_CONFIG);
g_AM29DL_16_BS_DeviceTable.EraseBlock(pBLOCK_CONFIG,MAC_address_area);
g_AM29DL_16_BS_DeviceTable.Write(pBLOCK_CONFIG,MAC_address_area,macLen,Buff,0x0);
download_printf( "Done\r\n" );

i = 0x100;
}
}
2) Set MAC address

Open
MicroFrameworkPK_v4_1\Solutions\SH7619_EVT\DeviceCode\Network\Ne
So that you can find g_NetworkConfig.

NETWORK_CONFIG g_NetworkConfig =
{
    { TRUE },
    1,   // interface count
    {
        0,   //0,
        //SOCK_MAKE_IP_ADDR(192,168, 84, 84), /*ip address */
        //SOCK_MAKE_IP_ADDR(255,255,255, 0), /*subnet mask*/
        //SOCK_MAKE_IP_ADDR(192,168, 84,251), /*gateway */
        //SOCK_MAKE_IP_ADDR(192,168, 84,200), /*dns1 */
        //SOCK_MAKE_IP_ADDR(192,168, 84,202), /*dns2 */
        SOCK_NETWORKCONFIGURATION_FLAGS_DHCP,
        SOCK_MAKE_IP_ADDR( 0, 0, 0, 0), /*ip address */
        SOCK_MAKE_IP_ADDR(255,255,255, 0), /*subnet mask*/
        SOCK_MAKE_IP_ADDR( 0, 0, 0, 0), /*gateway */
        SOCK_MAKE_IP_ADDR( 0, 0, 0, 0), /*dns1 */
        SOCK_MAKE_IP_ADDR( 0, 0, 0, 0), /*dns2 */
        // SOCK_NETWORKCONFIGURATION_FLAGS_DHCP |
        // SOCK_NETWORKCONFIGURATION_FLAGS_DYNAMIC_DNS |
        // SOCK_NETWORKCONFIGURATION_FLAGS_TYPE__set(SOCK_NETWORKCONFIGURATION_FL
        // SOCK_MAKE_IP_ADDR_LITTLEEND(192,168, 84, 9), /*ip address */
        // SOCK_MAKE_IP_ADDR_LITTLEEND(255,255,255, 0), /*subnet mask*/
        // SOCK_MAKE_IP_ADDR_LITTLEEND(192,168, 84,251), /*gateway */
        // SOCK_MAKE_IP_ADDR_LITTLEEND(192,168, 84,200), /*dns1 */
        // SOCK_MAKE_IP_ADDR_LITTLEEND(192,168, 84,202), /*dns2 */
        SOCK_NETWORKCONFIGURATION_INTERFACETYPE_ETHERNET,
        6,   /*mac address length*/
        { /*mac address*/
            0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
            0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
            0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
            0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
            0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
            0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
        }
    }
};
Put the MAC address into above red portion.
7. **How to deploy application**

.NET Micro Framework has the feature of developing Application using Visual Studio 2010.

In this section, the way to deploy application is explained by using Sample application.
7.1 Install .NET Micro Framework SDK

In order to use .NET Micro Framework with Visual Studio, you need to install .NET Micro Framework SDK (MicroFrameworkSDK.msi) into your PC.
7.2  Prepare Sample application

At first, you need to create the application. Please open Visual Studio.

i) Create new project

Select File menu -> New -> Project.

![Microsoft Visual Studio](image)

**Figure 7.1  Microsoft Visual Studio**

Please note, these screens are captured from Visual Studio2010 professional. You will see difference screens and menus in the case of Visual Studio 2010 Express Edition. But, you will find similar features in the menus of Visual Studio 2010 Express Edition.

ii) Select the Project type, Template and project name

Select below,

Project Type: Micro Framework

Template: Console Application
Name: HelloWorld

And press OK button.

![New Project](image)

**Figure 7.2  New Project**

Note depending upon your application please select the appropriate options. If you are creating a GUI based application it is advised to select “Window Application” option.
iii) Copy the source file of Sample program into this project

Right click on “HelloWorld” in the Solution Explorer and select Add -> Existing Item….

Figure 7.3 Microsoft Visual Studio
Select the below file,

C:\MicroFrameworkPK_v4_1\Product\Sample\HelloWorld\Main.cs

And press Add button.

Figure 7.4  Add Existing Item dialog box
iv) Remove unnecessary file

Remove Program.cs file from the HelloWorld project.
v) Build solution

Build Solution by selecting Build Solution menu,

![Microsoft Visual Studio Build Solution Menu](image)

**Figure 7.6 Microsoft Visual Studio**

Then build shall be completed successfully.
7.3 **Deployment and run the application**

In order to communicate with visual studio and your board, you need to download TinyCLR into the flash memory on the board.

It is assumed that you have build and downloaded TinyCLR into the flash memory on the board. If not, please build your Porting kit and downloading tinyclr.abs into the flash memory referring to section 4.

i) Change properties for current project.

Select Project menu -> HelloWorld Properties… so that you can modify properties.

![Figure 7.7  Microsoft Visual Studio](image)

Select “.NET Micro Framework” sheet and change deployment option to serial as shown below, save and close this window.
ii) Re-Build

In order to affect these changing, Re-build is required.

iii) Turn on the board

First, connected the serial terminal of board with that of PC using serial cable and then turn on the board.

iv) Deploying application

Select Build menu -> Deploy Solution.
Below message is shown up in the output window at the start of deployment process.

“Incrementally deploying assemblies to device”

When deployment gets finished without problem, you can see below message.

“Assemblies successfully deployed to device.”

If you have any trouble please make sure if you use correct flash memory configuration file.

v) Run the application

Select Debug menu -> Start Debugging so that you can see the application running and use break point via Visual Studio.

If the deployment fails, it might be caused by the unexpected data in the Deployment area in Flash memory. One of the way to avoid this is to erase all data in the deployment area using MFDeploy.exe. How to use MFDeploy.exe is described in Section 9.3.
7.4 Big-endian support

Normally, in the initial setting, the reference dlls are supporting Little-Endian. So it is recommended to change the reference dlls to for Big-Endian.

i) Remove the Reference for Little-Endian

Right Click the Reference which you want to remove as shown below (Microsoft.SPOT.Native) and select “Remove” option from the popup menu.

![Figure 7.10 Solution Explorer](image)

ii) Add the Reference for Big-Endian

Right Click on the “References” as shown below and select “Add reference...” option form the popup menu.

![Figure 7.11 Solution Explorer](image)
Then, “Add reference” window shall show up.

Select “Browse” tab and change “Look in” path to below mentioned path:

C:\Program Files\Microsoft .NET Micro Framework\v4.1\Assemblies\be

![Add Reference Window](image)

**Figure 7.12  Add Reference**

Then, select the dll which you want to add. (In this case, select Microsoft.SPOT.Native.dll.)
iii) Make sure that selected reference dll is correct.

Double Click the Reference which you want to see information.

![Solution Explorer](image)

**Figure 7.13  Solution Explorer**

An object browser tab / window shall be displayed. Verify that you see the correct path information for dll (Microsoft.SPOT.Native) as shown below.
Figure 7.14  Object Browser
8. **How to include application in TinyCLR**

The application can be included into tinyclr.abs. The simplest thing to do is just to add the PE files to your TinyCLR.proj file.

1) Build generic application in the Porting Kit

Change current folder to “C:\MicroFrameworkPK_v4_1’ and execute below command,

```
MSBUILD.EXE build.dirproj
```

2) Build an application without Visual Studio 2010.

Build your application by following command,

```
MSBUILD.EXE build.dirproj
```

For example, if you want to build HelloWorld Sample application, use following commands,

Change current folder to “C:\MicroFrameworkPK_v4_1\Product\Sample’ and execute below command,

```
MSBUILD.EXE build.dirproj
```

3) How to include application into tinyclr.abs

Please add the following red lines after the property section of the TinyCLR.proj file. You will have to make sure the .pe files have been built.

```xml
...<Import Condition="" Project="$(SPOCLIENT)\Framework\Features\Diagnostics.featureproj" />
<Import Condition="" Project="$(SPOCLIENT)\Framework\Features\Core.featureproj" />
<Import Condition="" Project="$(SPOCLIENT)\Framework\Features\Serialization.featureproj" />

<ItemGroup>
  <MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\mscorlib.pe"/>
  <MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.Native.pe"/>
  <MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.Net.pe"/>
</ItemGroup>
```

Change below red lines appropriate for your application
<ItemGroup>
  <MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.Graphics.pe"/>
  <MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.Hardware.pe"/>
  <MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.HelloWorld.pe"/>
  <MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\System.pe"/>
  <MMP_DAT_CreateDatabase Include="$(BUILD_TREE_CLIENT)\pe\$(ENDIANNESS)\Microsoft.SPOT.TinyCore.pe"/>
</ItemGroup>

<Import Project="$(SPOCLIENT)\tools\targets\Microsoft.SPOT.System.Interop.Settings" />
9. **MFDeploy Tool**

In Porting Kit, there is a useful Tool, MFDeploy.exe. Using this tool, you can make sure if TinyCLR works fine, see the Flash memory mapping information, and erase the data in Deployment area.
9.1 How to build and run MFDeploy.exe

Following is the way to build and run MFDeploy.exe.

1) Make sure the Timer driver, Power driver and the Serial driver work fine

Before using MFDeploy.exe, please make sure the Timer driver, Power driver and the Serial driver works fine using NativeSample.

2) Build MFDeply.exe

You can build MFDeploy.exe with the command “MSBUILD.EXE build.dirproj” under the C:\MicroFrameworkPK_v4_1 folder.

Using this command, you can build not only MDFelpoy.exe but also all the managed tests in the PK as well.

MFdeploy is located in

\BuildOutput\public\Debug\Server\dll\MFDeploy.exe.

3) Run the MFDeply.exe

i) Connect COM Port to the board

Before run the MFDeploy.exe, please make sure the COM port in your PC is not used by another application like Tera Term.

Then connect the COM port in your PC to the Serial connector on the board by Serial Cross Cable.

ii) Run the MFDeply.exe

Run the MFDeploy.exe by double-clicking it.

iii) Select the COM Port

Select the COM Port which you want to use for MFDeploy as shown below (List Box).
iii) Connect COM Port

Connect MFdeploy.exe to COM Port by selecting Target menu -> Connect option as shown below.
Figure 9.2 .NET Micro Framework Deployment Tool
9.2 How to make sure if TinyCLR works fine

If you can not deploy the application properly, “MFDeploy.exe” is a good tool to make sure if the TinyCLR works fine.

Turn on the board so that you can see like below log output.

Connecting to COM1...Connected

TinyCLR (Build 4.0.2037.0)
Starting...
Created EE.
Started Hardware.
No debugger!
Create TS.
Loading start at 95000, end b5f88
Attaching file.
Assembly: mscorlib (4.0.2037.0) (3572 RAM - 29944 ROM - 17631 METADATA)
  AssemblyRef = 0 bytes ( 0 elements)
  TypeRef = 0 bytes ( 0 elements)
  FieldRef = 0 bytes ( 0 elements)
  MethodRef = 0 bytes ( 0 elements)
  TypeDef = 1032 bytes ( 129 elements)
  FieldDef = 232 bytes ( 115 elements)
  MethodDef = 1448 bytes ( 724 elements)

  Attributes = 0 bytes ( 0 elements)
  TypeSpec = 16 bytes ( 4 elements)
  Resources = 232 bytes ( 29 elements)
  Resources Files = 16 bytes ( 2 elements)
  Resources Data = 437 bytes
  Strings = 967 bytes
  Signatures = 2015 bytes
  ByteCode = 10500 bytes
Attaching file.
Assembly: Microsoft.SPOT.Native (4.0.2037.0) (1064 RAM - 5752 ROM - 4159 METADATA)

  AssemblyRef = 4 bytes ( 1 elements)
  TypeRef = 80 bytes ( 20 elements)
  FieldRef = 0 bytes ( 0 elements)
  MethodRef = 60 bytes ( 15 elements)
  TypeDef = 328 bytes ( 41 elements)
  FieldDef = 132 bytes ( 65 elements)
  MethodDef = 216 bytes ( 108 elements)

  Attributes = 48 bytes ( 6 elements)
  TypeSpec = 0 bytes ( 0 elements)
Resources = 72 bytes ( 9 elements)
Resources Files = 8 bytes ( 1 elements)
Resources Data = 747 bytes
Strings = 207 bytes
Signatures = 587 bytes
ByteCode = 413 bytes

Attaching file.
Assembly: Microsoft.SPOT.Hardware (4.0.2037.0) (1752 RAM - 11404 ROM - 7365 METADATA)
AssemblyRef = 8 bytes ( 2 elements)
TypeDef = 124 bytes ( 31 elements)
FieldRef = 24 bytes ( 6 elements)
MethodRef = 120 bytes ( 30 elements)
TypeDef = 496 bytes ( 62 elements)
FieldDef = 176 bytes ( 88 elements)
MethodDef = 444 bytes ( 222 elements)
Attributes = 0 bytes ( 0 elements)
TypeSpec = 0 bytes ( 0 elements)
Resources = 0 bytes ( 0 elements)
Resources Files = 0 bytes ( 0 elements)
Resources Data = 0 bytes
Strings = 1329 bytes
Signatures = 1061 bytes
ByteCode = 2579 bytes

Attaching file.
Assembly: Microsoft.SPOT.Hardware.SerialPort (4.0.2037.0) (508 RAM - 3440 ROM - 1527 METADATA)
AssemblyRef = 12 bytes ( 3 elements)
TypeDef = 96 bytes ( 24 elements)
FieldRef = 0 bytes ( 0 elements)
MethodRef = 80 bytes ( 20 elements)
TypeDef = 16 bytes ( 2 elements)
FieldDef = 32 bytes ( 16 elements)
MethodDef = 92 bytes ( 46 elements)
Attributes = 0 bytes ( 0 elements)
TypeSpec = 0 bytes ( 0 elements)
Resources = 0 bytes ( 0 elements)
Resources Files = 0 bytes ( 0 elements)
Resources Data = 0 bytes
Strings = 667 bytes
Signatures = 239 bytes
ByteCode = 1118 bytes

Attaching file.
Assembly: Microsoft.SPOT.IO (4.0.2037.0) (716 RAM - 4432 ROM - 2459 METADATA)
AssemblyRef = 12 bytes ( 3 elements)
TypeDef = 72 bytes ( 18 elements)
| FieldRef   | 0 bytes (0 elements) |
| MethodRef | 96 bytes (24 elements) |
|TypeDef    | 120 bytes (15 elements) |
|FieldDef   | 68 bytes (34 elements) |
|MethodDef  | 140 bytes (70 elements) |

 Attributes   | 0 bytes (0 elements) |
 TypeSpec     | 0 bytes (0 elements) |
 Resources    | 0 bytes (0 elements) |
 Resources Files | 0 bytes (0 elements) |
 Resources Data | 0 bytes |
 Strings   | 646 bytes |
 Signatures | 335 bytes |
 ByteCode   | 1199 bytes |

 Attaching file.

 Assembly: System.IO (4.0.2037.0) (1548 RAM - 13264 ROM - 5862 METADATA)
 | AssemblyRef | 8 bytes (2 elements) |
 | TypeRef     | 168 bytes (42 elements) |
 | FieldRef    | 36 bytes (9 elements) |
 | MethodRef   | 392 bytes (98 elements) |
 |TypeDef      | 144 bytes (18 elements) |
 |FieldDef     | 76 bytes (37 elements) |
 |MethodDef    | 392 bytes (195 elements) |

 Attributes   | 0 bytes (0 elements) |
 TypeSpec     | 8 bytes (2 elements) |
 Resources    | 0 bytes (0 elements) |
 Resources Files | 0 bytes (0 elements) |
 Resources Data | 0 bytes |
 Strings   | 356 bytes |
 Signatures | 790 bytes |
 ByteCode   | 6919 bytes |

 Attaching file.

 Assembly: Microsoft.SPOT.Graphics (4.0.2037.0) (388 RAM - 2268 ROM - 1357 METADATA)
 | AssemblyRef | 8 bytes (2 elements) |
 | TypeRef     | 24 bytes (6 elements) |
 | FieldRef    | 0 bytes (0 elements) |
 | MethodRef   | 20 bytes (5 elements) |
 |TypeDef      | 40 bytes (5 elements) |
 |FieldDef     | 16 bytes (8 elements) |
 |MethodDef    | 96 bytes (48 elements) |

 Attributes   | 0 bytes (0 elements) |
 TypeSpec     | 0 bytes (0 elements) |
 Resources    | 0 bytes (0 elements) |
 Resources Files | 0 bytes (0 elements) |
 Resources Data | 0 bytes |
 Strings   | 537 bytes |
 Signatures | 293 bytes |
 ByteCode   | 242 bytes |
Attaching file.
Assembly: Microsoft.SPOT.TinyCore (4.0.2037.0) (5080 RAM - 61564 ROM - 23446 METADATA)

AssemblyRef = 16 bytes (4 elements)
TypeDef = 224 bytes (56 elements)
FieldRef = 52 bytes (13 elements)
MethodRef = 456 bytes (114 elements)
TypeDef = 1104 bytes (138 elements)
FieldDef = 728 bytes (363 elements)
MethodDef = 1576 bytes (787 elements)

Attributes = 0 bytes (0 elements)
TypeSpec = 4 bytes (1 elements)
Resources = 0 bytes (0 elements)
Resources Files = 0 bytes (0 elements)
Resources Data = 0 bytes
Strings = 12916 bytes
Signatures = 3122 bytes
ByteCode = 25075 bytes

Attaching file.
Assembly: Microsoft.SPOT.Time (4.0.2037.0) (508 RAM - 2976 ROM - 1552 METADATA)

AssemblyRef = 12 bytes (3 elements)
TypeDef = 60 bytes (15 elements)
FieldRef = 0 bytes (0 elements)
MethodRef = 36 bytes (9 elements)
TypeDef = 96 bytes (12 elements)
FieldDef = 40 bytes (20 elements)
MethodDef = 84 bytes (41 elements)

Attributes = 0 bytes (0 elements)
TypeSpec = 0 bytes (0 elements)
Resources = 0 bytes (0 elements)
Resources Files = 0 bytes (0 elements)
Resources Data = 0 bytes
Strings = 895 bytes
Signatures = 220 bytes
ByteCode = 403 bytes

Loading Deployment Assemblies.
Resolving.

Total: (12884 RAM - 135044 ROM - 65358 METADATA)

AssemblyRef = 76 bytes (19 elements)
TypeDef = 848 bytes (212 elements)
FieldRef = 112 bytes (28 elements)
MethodRef = 1260 bytes (315 elements)
TypeDef = 3376 bytes (422 elements)
FieldDef = 1500 bytes (746 elements)
MethodDef = 4488 bytes (2241 elements)
DebuggingInfo = 2252 bytes

Attributes = 48 bytes (6 elements)
TypeSpec = 28 bytes (7 elements)
Resources Files = 72 bytes (3 elements)
Resources = 304 bytes (38 elements)
Resources Data = 1184 bytes
Strings = 18520 bytes
Signatures = 8662 bytes
ByteCode = 48448 bytes

GC: 1msec 15276 bytes used, 4178820 bytes available
Type 0F (STRING ): 24 bytes
Type 15 (FREEBLOCK ): 4178820 bytes
Type 17 (ASSEMBLY ): 15180 bytes
Type 34 (APPDOMAIN_HEAD ): 72 bytes

Total: (12884 RAM - 135044 ROM - 65358 METADATA)
AssemblyRef = 76 bytes (19 elements)
TypeDef = 3376 bytes (422 elements)
FieldDef = 1500 bytes (746 elements)
MethodDef = 4488 bytes (2241 elements)

DebuggingInfo = 2252 bytes

Attributes = 48 bytes (6 elements)
TypeSpec = 28 bytes (7 elements)
Resources Files = 72 bytes (3 elements)
Resources = 304 bytes (38 elements)
Resources Data = 1184 bytes
Strings = 18520 bytes
Signatures = 8662 bytes
ByteCode = 48448 bytes

Ready.
Cannot find any entrypoint!
Done.
Waiting for debug commands...

After the above messages, you should try and connect from MFdeploy by clicking “Ping” button from MFdeploy. If you see that the TinyCLR responds, then it means that the TinyCLR is up and running.

Pinging... TinyCLR
9.3 Erase data in the deployment area

Following is the way to erase data in the deployment area.

1) Connect the Target Platform referring section 9.1.

2) Press Erase button so that erasing starts.

![Figure 9.3 .NET Micro Framework Deployment Tool](image)

3) If below error shows up, don’t mind.

![Figure 9.4 Error message](image)
4) Following message shows up when the data in deployment area has erased.

![Figure 9.5 .NET Micro Framework Deployment Tool](image)

**Figure 9.5 .NET Micro Framework Deployment Tool**
10. **Wiring arrangement**

In this section, the wiring arrangement for hardware for existing LCD driver and Keypad driver is described.

1) Wiring arrangement for LDC panel

The 132 x 176 TFT LCD Display (HD66773R) is interfaced to area CS6B of the SH7619 Bus State Controller (BSC). 3.3V power is provided by the SH7619 board, while 5V power is provided by separate DC power supply connected to the LCD Expansion board. A 40-pin header connector is used for the LCD module interface; the pin assignment for the 40-pin header is shown below.

<table>
<thead>
<tr>
<th>Pin No.</th>
<th>Signal Name</th>
<th>I/O</th>
<th>Pin No.</th>
<th>Signal Name</th>
<th>I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GND</td>
<td>—</td>
<td>21</td>
<td>D3</td>
<td>I/O</td>
</tr>
<tr>
<td>2</td>
<td>iRESET</td>
<td>OUT</td>
<td>22</td>
<td>D2</td>
<td>I/O</td>
</tr>
<tr>
<td>3</td>
<td>3.3V supply</td>
<td>—</td>
<td>23</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>NC</td>
<td>—</td>
<td>24</td>
<td>D1</td>
<td>I/O</td>
</tr>
<tr>
<td>5</td>
<td>NC</td>
<td>—</td>
<td>25</td>
<td>D0</td>
<td>I/O</td>
</tr>
<tr>
<td>6</td>
<td>D15</td>
<td>I/O</td>
<td>26</td>
<td>D15</td>
<td>—</td>
</tr>
<tr>
<td>7</td>
<td>D14</td>
<td>I/O</td>
<td>27</td>
<td>/LCD-RD</td>
<td>OUT</td>
</tr>
<tr>
<td>8</td>
<td>GND</td>
<td>—</td>
<td>28</td>
<td>/LCD-WR</td>
<td>OUT</td>
</tr>
<tr>
<td>9</td>
<td>D13</td>
<td>I/O</td>
<td>29</td>
<td>LCD-RS1</td>
<td>OUT</td>
</tr>
<tr>
<td>10</td>
<td>D12</td>
<td>I/O</td>
<td>30</td>
<td>LCD-RS2</td>
<td>OUT</td>
</tr>
<tr>
<td>11</td>
<td>D11</td>
<td>I/O</td>
<td>31</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>12</td>
<td>D10</td>
<td>I/O</td>
<td>32</td>
<td>3.3V supply</td>
<td>—</td>
</tr>
<tr>
<td>13</td>
<td>GND</td>
<td>—</td>
<td>33</td>
<td>5V supply</td>
<td>—</td>
</tr>
<tr>
<td>14</td>
<td>D9</td>
<td>I/O</td>
<td>34</td>
<td>5V supply</td>
<td>—</td>
</tr>
<tr>
<td>15</td>
<td>D8</td>
<td>I/O</td>
<td>35</td>
<td>GND</td>
<td>—</td>
</tr>
<tr>
<td>16</td>
<td>D7</td>
<td>I/O</td>
<td>36</td>
<td>/LCD-CS1</td>
<td>OUT</td>
</tr>
<tr>
<td>17</td>
<td>D6</td>
<td>I/O</td>
<td>37</td>
<td>/LCD-CS2</td>
<td>OUT</td>
</tr>
<tr>
<td>18</td>
<td>GND</td>
<td>—</td>
<td>38</td>
<td>LCD-TMG1</td>
<td>IN</td>
</tr>
<tr>
<td>19</td>
<td>D5</td>
<td>I/O</td>
<td>39</td>
<td>LCD-TMG2</td>
<td>IN</td>
</tr>
<tr>
<td>20</td>
<td>D4</td>
<td>I/O</td>
<td>40</td>
<td>LCD-DIR</td>
<td>OUT</td>
</tr>
</tbody>
</table>

**Figure 10.1  LCD header pin assignment**
Figure 10.2  Schematic of LCD interface
2) Wiring arrangement for GPIO for Button Keypad

In this Porting Kit, it is assumed that a 25-button keypad matrix is connected to ten GPIOs of the SH7619. Five GPIOs (PC08, PC10, PC09, PC11, and PC13) should be configured for output. The other five GPIOs (PC00, PC01, PC02, PC03, and PC17) should be configured for input and they have pull-up resistors meaning the buttons are active low.

![GPIO button configuration and perspective orientation](image)

**Figure 10.3** GPIO button configuration and perspective orientation
3) Wiring arrangement for secondary Serial Interface
Figure 10.4  Schematic of Secondary Serial Interface