logic IO
The M2M Enabler

Application Note
DynDNS support
Version 1.01

This application note describes how to include support for DynDNS in VPL applications.

DynDNS is a service provided by www.dyndns.com that allows binding of a canonical symbolic name, such as **test.rtcu.dk** to an IP address such as **210.87.12.124**. The DynDNS service is especially designed for scenarios where the IP-address is dynamically assigned to a router or firewall. The service can however also be applied to an RTCU unit operating with GPRS, although using the RTCU Gateway concept is the preferred way of communication.

When the network topology assigns a global dynamic IP-address to the RTCU unit communication with the unit can be accomplished using the DynDNS service to keep track of the dynamically allocated IP-address. In the cases where the IP-address allocated is inside of a firewall this method can not be used.

The software for the support is a function that performs an update of the registered IP address. The update transaction is based on basic HTTP, and is described on the DynDNS web page.

The DynDNSUpdate function handles the entire sequence, from building the HTTP request, over opening a socket to the DynDNS server and sending the request, to waiting for the reply. All the required parameters are present as Input variables in the function, and the reply from DynDNS is returned for possible further handling.

To include the support, add the 'DynDNS.inc' to your VPL application and call the DynDNSUpdate function after the unit is connected to GPRS.

A complete example project is included to demonstrate how updating a DynDNS account can be done. This example is ready to run from a RTCU or through the simulator in the RTCU IDE. The account updated is the official developer account which is free to use.

Note that most GSM providers blocks incoming socket connections (connecting to the RTCU from another source), and that the provider must be contacted to open for this.

On the following pages the source code for updating the client can be seen.

Logic IO Aps
Holmboes Allé 14
8700 Horsens
Denmark

Ph: (+45) 7625 0210
Fax: (+45) 7625 0211
Email: info@logicio.com
www.logicio.com

Appendix 1: Source code listing

```
//------------------------------------------------------------------------------
// Updates DynDNS.
//
// Input:
//    userid    - Username for DynDNS account.
//    password  - Password for DynDNS account.
//    domain    - The domain name for the unit.
//    agent     - The user agent id string for the unit.
//
// Returns:
//    String containing the reply from DynDNS.
//------------------------------------------------------------------------------
FUNCTION DynDNSUpdate : STRING;
VAR_INPUT
   userid    : STRING;
   password  : STRING;
   domain    : STRING;
   agent     : STRING;
END_VAR;
VAR
   // HTTP GET command
   Line1     : STRING;
   Line2     : STRING;
   Line3     : STRING;
   Line4     : STRING;
   Line5     : STRING;

   // Socket
   ip        : DINT;
   ipaddress : STRING;
   id        : SINT;
   timeout   : DINT;
   rc        : INT;
   TempStr   : STRING;

END_VAR;

   IF (NOT gprsConnected()) THEN
      DebugMsg(message := "[DynDNS] Socket - Not connected to GPRS");
      RETURN;
   END_IF;

   // Get IP address
   ip        := sockGetLocalIP();
   ipaddress := sockIPToName(ip := ip);

   // Build update string
   Line1 := "GET /nic/update?hostname=" + domain + "&myip=" + ipaddress;
   Line1 := Line1 + "&wildcard=NOCHG&mx=NOCHG&backmx=NOCHG HTTP/1.0$N";

   Line2 := "Host: members.dyndns.org$N";
   // Encode userid-password
   Line3 := "Authorization: Basic " + strEncode64(str := userid + ":" + password) + "$N";
   Line4 := "User-Agent: " + agent + "$N";
   Line5 := "$N";

   // Lookup the IP address of the host (DNS lookup)
   ip := sockIPFromName(str := "members.dyndns.org");
```


Logic IO Aps
Holmboes Allé 14
8700 Horsens
Denmark

Ph: (+45) 7625 0210
Fax: (+45) 7625 0211
Email: info@logicio.com
www.logicio.com

```
   // Connect to the host
   id := sockConnect(ip := ip, port := 8245);
   IF (id > 0) THEN
      // Configure function blocks
      DynDNS_Socket.id  := id;
      DynDNS_SockReceive.id := id;

      // Wait for connection
      timeout := clockNow() + 10;  // Wait 10 seconds
      DynDNS_Socket();
      WHILE ((NOT DynDNS_Socket.Connected) AND (timeout > clockNow())) DO
         Sleep(delay:=100);
         DynDNS_Socket();
      END_WHILE;

      IF DynDNS_Socket.Connected THEN
         DebugMsg(message:="[DynDNS] Socket - Connected to host");

         // Send command
         strToMemory(dst:=ADDR(DynDNS_Buffer), str:=Line1, len:=strLen(str:=Line1));
         rc := sockSend(id:=id, data:=ADDR(DynDNS_Buffer), size:=strLen(str:=Line1));
         DebugFmt(message:="[DynDNS] Socket - Send Line1 rc=\1",v1:=rc);

         strToMemory(dst:=ADDR(DynDNS_Buffer), str:=Line2, len:=strLen(str:=Line2));
         rc := sockSend(id:=id, data:=ADDR(DynDNS_Buffer), size:=strLen(str:=Line2));
         DebugFmt(message:="[DynDNS] Socket - Send Line2 rc=\1",v1:=rc);

         strToMemory(dst:=ADDR(DynDNS_Buffer), str:=Line3, len:=strLen(str:=Line3));
         rc := sockSend(id:=id, data:=ADDR(DynDNS_Buffer), size:=strLen(str:=Line3));
         DebugFmt(message:="[DynDNS] Socket - Send Line3 rc=\1",v1:=rc);

         strToMemory(dst:=ADDR(DynDNS_Buffer), str:=Line4, len:=strLen(str:=Line4));
         rc := sockSend(id:=id, data:=ADDR(DynDNS_Buffer), size:=strLen(str:=Line4));
         DebugFmt(message:="[DynDNS] Socket - Send Line4 rc=\1",v1:=rc);

         strToMemory(dst:=ADDR(DynDNS_Buffer), str:=Line5, len:=strLen(str:=Line5));
         rc := sockSend(id:=id, data:=ADDR(DynDNS_Buffer), size:=strLen(str:=Line5));
         DebugFmt(message:="[DynDNS] Socket - Send Line5 rc=\1",v1:=rc);

         // Wait for reply
         DynDNS_SockReceive.data    := ADDR(DynDNS_Buffer);
         DynDNS_SockReceive.maxsize := SIZEOF(DynDNS_Buffer);
         timeout := clockNow() + 10;  // Wait 10 seconds
         // expect server to close socket when message has bee received.
         WHILE ((DynDNS_Socket.connected OR DynDNS_SockReceive.ready) AND (timeout >
clockNow()) ) DO
            DynDNS_Socket();
            DynDNS_SockReceive();
            IF DynDNS_SockReceive.ready THEN
               TempStr := TempStr +
strFromMemory(src:=ADDR(DynDNS_Buffer),len:=DynDNS_SockReceive.size);
            END_IF;
         END_WHILE;

         // get the result of the update
         DynDNSUpdate := strToken(str:=TempStr, delimiter:="$N$N", index:=2);
         DebugMsg(message:= "[DynDNS] Reply = " + DynDNSUpdate);
         IF (strLen(str := DynDNSUpdate) = 0) THEN
            // Timeout waiting for reply
            IF (strLen(str := TempStr) > 0) THEN
```


Logic IO Aps
Holmboes Allé 14
8700 Horsens
Denmark

Ph: (+45) 7625 0210
Fax: (+45) 7625 0211
Email: info@logicio.com
www.logicio.com

```
            DebugMsg(message := "[DynDNS] Socket - Timeout waiting for reply got :");
            DebugMsg(message := "                   '" + Tempstr +"'");
         ELSE
            DebugMsg(message:="[DynDNS] Socket - Timeout waiting for reply");
         END_IF;
      END_IF;
   ELSE
      // Timeout waiting for connection
      DebugMsg(message:="[DynDNS] Socket - Timeout waiting for connection");
   END_IF;

   // Close socket after use
   sockDisconnect(id:=id);
ELSE
   // Failed to open socket!
   DebugMsg(message:="[DynDNS] Socket - Failed to open");
END_IF;
END_FUNCTION
```


Logic IO Aps
Holmboes Allé 14
8700 Horsens
Denmark

Ph: (+45) 7625 0210
Fax: (+45) 7625 0211
Email: info@logicio.com
www.logicio.com