INTRODUCTION

Technologies that allow both wireless and wired systems to communicate with other devices of the same ability are referred to as Machine-to-Machine (M2M). M2M uses a device to capture an event, which is then relayed through a network to an application that translates the event into meaningful information. A common application of M2M is fleet management, where vehicle tracking is wirelessly transmitted to a central monitoring office over cellular networks.

There are many popular M2M applications, one of which is a utility meter. One of the major benefits of a M2M-based utility meter over a traditional one, is immediate operational efficiency from reading and programming meters remotely, which eliminates the need to physically visit the meter.

Another application becoming more popular with M2M technology is in-car GPS navigation. With this technology, consumers now have a complete GPS navigation system in their vehicles. This technology can be used to track a driver’s current location, or provide a map for directions. Also, the consumer can make an emergency call from the same device that is doing the tracking. Businesses can use this technology for parking lots to know how long a vehicle has been parked and to charge it accordingly.

The Microchip M2M PICtail™ Plus Daughter Board (referred to as the M2M Board) developed by u-blox AG, was designed to connect directly to the PICtail™ interface of the Multimedia Expansion Board (MEB), but can also be used with any PIC32 microcontroller.

This application note describes a reference design that enables the implementation of GSM/GPRS/GPS connectivity using a PIC32 microcontroller (MCU), the M2M Board, and the MEB.

Feature Overview

The M2M PICtail Plus Daughter Board contains many features, including GSM, GPRS, and GPS.

- Global System for Mobile Communication (GSM)
  GSM is a popular world-wide standard for mobile telephone systems. GSM includes technologies in both signaling and speech channels, which are digital; therefore, GSM is considered a Second Generation (i.e., 2G) mobile phone system. This facilitates the wide-spread implementation of data communication applications into the system. GSM also implements a Short Message Service (SMS), called text messaging.

- General Packet Radio Service (GPRS)
  GPRS is a service on 2G and 3G cellular communication systems (GSM). GPRS provides data rates of 56-114 kbps, which provides users with the capability to connect to the Internet.

- Global Positioning System (GPS)
  GPS is a space-based navigation system that provides reliable location and time information in all weather conditions and at all times, and anywhere on or near the Earth when and where there is an unobstructed line of sight to four or more GPS satellites. It is freely accessible by anyone with a GPS receiver.

Functionality

The main functionality of the M2M Board is accomplished using two communications modules from u-blox A, which is a company that specializes in GSM/ GPS ICs. For more information, visit www.u-blox.com.

The M2M Board was designed to connect to Microchip’s MEB. The MEB uses Microchip’s PIC32 starter kit collection as the primary controller source. This suite makes it easy to start and implement embedded controller projects due to its:

- Built-in debugger
- USB power source
- On-board header for easy attachment to PCBs
- PIC32 device with high-speed performance and no peripheral loss
HARDWARE DESCRIPTION

As previously mentioned, the M2M Board is built around two controller modules, the LEON-G200 and the NEO-6Q, which are available from u-blox AG.

The LEON-G200 is a Quad Band GSM/GPRS data and voice module. Communications to the module are through AT commands. The UART module on the PIC32 device handles the AT commands. The LEON-G200 handles the GPS communications to the NEO-6Q module. The module also contains 1 MB of non-volatile memory that can be used for storing local or Internet files.

The NEO-6Q GPS module uses the u-blox six-positioning engine for its GPS positions support. In this hardware setup, it acts as a slave to the LEON-G200, but can be a stand-alone module with its own set of AT commands. For more information on these two modules and a list of AT commands, visit www.u-blox.com.

A block diagram of the reference design is provided in Figure 1.

The M2M Board connects to expansion slot header J5 of the MEB. Figure 1 illustrates the connector pins. Descriptions of each pin are listed in Table 1.
The M2M board also implements the following:

• SAW filter
  The filter is located in front of the LNA to improve the GPS interference immunity (outband interference from collocation of near field Wireless communication). The SAW-LNA-SAW chain is implemented for best immunity and performance.

• SIM holder
  The holder is used for the SIM card, which enables GPRS/GSM communication evaluation.

• RF SMA Connector
  The connector is located in front of antenna detection and switch circuitry, which allows automatic hardware detection and connection onto a GPS external antenna. Using an external GPS external antenna is optional and allows for better GPS performance under poor GPS conditions.

Figure 3 provides a detailed diagram of the M2M Board.
As seen in Table 1, most of the communication to the LEON-G200 is handled through the UART. The MEB communicates to the M2M Board through the UART. The baud rate can be an auto baud rate, and most of the common baud rates are acceptable.

The PIC32 family of devices offer the same peripherals as seen on previous Microchip devices.

See “References” for links to information on the hardware and the PIC32 device used in this reference design.

### DEMONSTRATION DESCRIPTION

This section describes in detail what is contained in the GSM/GPRS/GPS demonstration. Some of the images may contain different graphics than those shown in this application note, but the basic functionality of the demonstration is still present.

The MEB contains a 3.2" QVGA touch screen display, making the demonstration easy to follow, and shows how each service is set up. The Debug output of the PIC32 Starter Kit can be used to see which u-blox commands are being used throughout the demonstration.

The demonstration starts with the initialization of all GSM/GPS/GPRS services. Notice in Figure 4 that most buttons seem disabled except for HELP and NEXT. Selecting NEXT initializes the M2M Board setup. The M2M Board requires a SIM card with a data plan to be inserted into the back of it in order for the GSM/GPRS demonstrations to function. The initialization step has three screens that initialize GPS, GPRS, and GPS technologies, one at a time. Some technologies require some time (10-30 seconds) for their service provider to respond.

The GSM demonstration shows the current signal strength and the service provider. With this demonstration, the user can also send a text message by selecting the SMS button, as shown in Figure 5. Once SMS is selected, the user will be taken to a screen where they can enter a phone number. Once the number is entered and a SIM card is inserted into the M2M Board, a message from the M2M Board will be sent to that phone number.

By default, the M2M demonstration has SMS receiving turned ON. This means that if a text message is sent to the board, the demo will pause and show the number and message received, and then return to the demo.

### TABLE 1: MEB CONNECTOR PIN DESCRIPTIONS

<table>
<thead>
<tr>
<th>Connector Pin Name</th>
<th>MEB Pin Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI</td>
<td>WIFI_SDO</td>
<td>UART ring indicator</td>
</tr>
<tr>
<td>DTR</td>
<td>WIFI_SDI</td>
<td>UART data terminal ready</td>
</tr>
<tr>
<td>DSR</td>
<td>WIFI_SCK</td>
<td>UART data set ready</td>
</tr>
<tr>
<td>RXD</td>
<td>SDI3A</td>
<td>UART receive data</td>
</tr>
<tr>
<td>TXD</td>
<td>SDO3A</td>
<td>UART transmit data</td>
</tr>
<tr>
<td>RTS</td>
<td>SCK3A</td>
<td>UART ready to send</td>
</tr>
<tr>
<td>CTS</td>
<td>SS3A/RF12</td>
<td>UART clear to send</td>
</tr>
<tr>
<td>DCD</td>
<td>WIFI_CS</td>
<td>UART data carrier detect</td>
</tr>
<tr>
<td>PWR_ON</td>
<td>C2OUT/AN9</td>
<td>Turns the device on/off</td>
</tr>
<tr>
<td>RESET_N</td>
<td>RA10</td>
<td>Holds the device in reset</td>
</tr>
</tbody>
</table>

As seen in Table 1, most of the communication to the LEON-G200 is handled through the UART.

The MEB communicates to the M2M Board through the UART. The baud rate can be an auto baud rate, and most of the common baud rates are acceptable.

The PIC32 family of devices offer the same peripherals as seen on previous Microchip devices.
The GPRS demo (see Figure 6) shows to which IP address the device is connected. This screen initiates the basic Internet connection needed to create demonstrations, such as an e-mail service, simple Web browsing, and any other 2G phone feature involving an Internet connection.

The GPS demonstration shows the current longitude and latitude coordinates, as shown in Figure 7.

The Email button will become enabled when an active SIM card is present with data services available. By selecting the Email button, the user will be sent to a screen where an e-mail address can be entered, and if an e-mail server has been set up correctly, an e-mail can be sent. Refer to the `BYTE acdEmailSetup()` function to change the email server name. The M2M board does not support SSL-based SMTP servers. The Map button will remain enabled until GPS data is available.

Note: GPS data is not ready until the LED labeled TIMEPULSE on the M2M Board is blinking green.
GSM/GPRS/GPS STACK
DESCRIPTION

The GSM/GPRS/GPS stack was built around u-blox communication commands. The basic commands can be found on the u-blox AG website at: www.u-blox.com.

This section describes in detail the high-level commands listed in Table 2, which are needed to interface with the GSM/GPRS/GPS stack. Detailed descriptions of each function are provided following the table.

### TABLE 2: ENUMERATOR AND FUNCTION DESCRIPTIONS

<table>
<thead>
<tr>
<th>Structure</th>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UBX_STATUS</td>
<td>UBX_S_SYSTEM_CONFIGURED</td>
<td>u-blox high-level interface is configured.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_SYSTEM_NOT_CONFIGURED</td>
<td>u-blox high-level interface is not configured.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_BOARD_POWERED</td>
<td>C16-G26Q plug-in board is powered-on.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_BOARD_NOT_POWERED</td>
<td>C16-G26Q plug-in board is powered-off.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_BOARD_PRESENT_LEONG200</td>
<td>C16-G26Q plug-in recognized as LEON-G200.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_BOARD_PRESENT</td>
<td>C16-G26Q plug-in board is present.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_BOARD_NOT_PRESENT</td>
<td>C16-G26Q plug-in board is not present.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_SIM_PRESENT</td>
<td>SIM card is present.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_SIM_NOT_PRESENT</td>
<td>SIM card is not present.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_PIN_ENABLED</td>
<td>PIN on SIM card is enabled.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_PIN_NOT_ENABLED</td>
<td>PIN on SIM card is not enabled.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GSM_CONFIGURED</td>
<td>GSM module is configured.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GSM_NOT_CONFIGURED</td>
<td>GSM module is not configured.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GSM_NETWORK_REGISTERED</td>
<td>GSM network is registered.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GSM_NETWORK_NOT_REGISTERED</td>
<td>GSM network is not registered.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_PDP_CONFIGURED</td>
<td>GPRS is configured.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_PDP_NOT_CONFIGURED</td>
<td>GPRS is not configured.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_PDP_SERVICE_REGISTERED</td>
<td>GPRS service is available.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_PDP_SERVICE_NOT_REGISTERED</td>
<td>GPRS service is not available.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GPS_CONFIGURED</td>
<td>GPS is configured.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GPS_NOT_CONFIGURED</td>
<td>GPS is not configured.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GPS_POWERED</td>
<td>GPS is powered on.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GPS_NOT_POWERED</td>
<td>GPS is powered off.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GPS_Assist_LOCAL</td>
<td>GPS with local aiding.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GPS_Assist_NONE</td>
<td>GPS without local aiding.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GPS_Assist_OFFLINE</td>
<td>GPS AssistNow is off-line.</td>
</tr>
<tr>
<td></td>
<td>UBX_S_GPS_Assist_ONLINE</td>
<td>GPS AssistNow is on-line.</td>
</tr>
</tbody>
</table>

| UBX_ERROR                  | UBX_E_OK                                | Success.                                              |
|                            | UBX_E_ERROR                             | Error, handling is required.                          |
|                            | UBX_E_WARNING                           | Warning, can be ignored.                              |

| UBX_GPS_ASSIST_MODE        | UBX_GPS_ASSIST_LOCAL                    | Power-on GPS with local aiding (use GSM cell info where possible). |
|                            | UBX_GPS_ASSIST_OFFLINE                 | Power-on GPS with AssistNow off-line (use 14 days almanac).     |
|                            | UBX_GPS_ASSIST_ONLINE                  | Power-on GPS with AssistNow on-line.                    |

| HTTP_REQUESTS             | HEAD                                    | Head command.                                         |
|                           | GET                                     | Get command.                                          |
|                           | DELETE                                  | Delete command.                                       |
|                           | PUT                                     | Put command.                                          |
|                           | POST_FILE                               | Post file command.                                    |
|                           | POST_DATA                               | Post data command.                                    |
Stack API Functions

UBX_ERROR ubxConfigureBoard(void)

Description
This function configures the C16-G26Q plug-in board.

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

Example
See Example 1.

UBX_ERROR ubxConfigureGps(void)

Description
This function configures the GPS module.

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

Example
See Example 3.

UBX_ERROR ubxConfigureGsm(void)

Description
This function configures the GSM network.

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

Example
See Example 2.
UBX_ERROR ubxConfigurePdp(char *apn)

Description
This function configures the GPRS service depending on the operator’s access point name.

Parameters
[in]  apn  Operator's Access Point Name (APN)

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

Example
See Example 4.

UBX_ERROR ubxConfigureSystem(UINT32 freq)

Description
This function configures the u-blox high-level interface.

Precondition
u-blox high-level interface should not be already configured

Parameters
[in]  freq  System (core) frequency

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

Example
#include <plib.h>
#include "libubx.h"
#include "libp32.h"

UBX_ERROR rc;
rc = ubxConfigureSystem(SYSTEM_FREQ);
if (rc != UBX_E_OK)
{
    return;
}

DBPRINTF("System core is running at %d Hz\n", p32GetSystemClock());
DBPRINTF("Peripheral bus is running at %d Hz\n", p32GetPeripheralBusClock());
**UBX_ERROR ubxGetBoardPowerStatus(void)**

**Description**
This function returns the status of the C16-G26Q plug-in board power.

**Returns**
- UBX_S_BOARD_POWERED
- UBX_S_BOARD_NOTPOWERED

**UBX_STATUS ubxGetBoardStatus(void)**

**Description**
This function returns the status of the C16-G26Q plug-in board.

**Returns**
- UBX_S_BOARD_PRESENT_LONG200
- UBX_S_BOARD_PRESENT
- UBX_S_BOARD_NOT_PRESENT

**UBX_STATUS ubxGetGpsAssistStatus(void)**

**Description**
This function returns the current aiding mode of the GPS module.

**Returns**
- UBX_S_GPS_ASSIST_LOCAL
- UBX_S_GPS_ASSIST_NONE
- UBX_S_GPS_ASSIST_OFFLINE
- UBX_S_GPS_ASSIST_ONLINE

**UBX_STATUS ubxGetGpsPowerStatus(void)**

**Description**
This function returns the power status of the GPS module.

**Returns**
- UBX_S_GPS_POWERED
- UBX_S_GPS_NOT_POWERED
**UBX_STATUS ubxGetGpsStatus(void)**

**Description**
This function returns the status of the GPS module.

**Returns**
- UBX_S_GPS_CONFIGURED
- UBX_S_GPS_NOT_CONFIGURED

**UBX_ERROR ubxGetGsmNetworkOperator(char* netop)**

**Description**
This function returns the name of the GSM network operator.

**Parameters**
- [out] `netop` GSM network operator, null-terminated string

**Returns**
- UBX_E_OK
- UBX_E_ERROR
- UBX_E_WARNING

**UBX_ERROR ubxGetGsmNetworkSignal(UINT32 *netss, char *netss_text)**

**Description**
This function returns the signal strength of the GSM network.

**Parameters**
- [out] `netss` GSM network signal strength, integer [0...5]
- [out] `netss_text` GSM network signal strength, null-terminated string

**Returns**
- UBX_E_OK
- UBX_E_ERROR
- UBX_E_WARNING
UBX_STATUS ubxGetGsmNetworkStatus(void)

Description
This function returns the status of the GSM network registration.

Returns
• UBX_S_GSM_NETWORK_REGISTERED
• UBX_S_GSM_NETWORK_NOT_REGISTERED

Example
See Example 2.

UBX_STATUS ubxGetGsmStatus(void)

Description
This function returns the status of the GSM network.

Returns
• UBX_S_GSM_CONFIGURED
• UBX_S_GSM_NOT_CONFIGURED

Example
See Example 2.

UBX_ERROR ubxGetPdpServiceAddress(char *ipadd)

Description
This function returns the IP address from GPRS service (current context).

Parameters
[out] ipadd GPRS service address, null-terminated string

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING
UBX_STATUS ubxGetPdpServiceStatus(void)

Description
This function returns the status of the GPRS services.

Returns
- UBX_S_PDP_SERVICE_REGISTRED
- UBX_S_PDP_SERVICE_NOT_REGISTRED

UBX_STATUS ubxGetPdpStatus(void)

Description
This function returns the status of the GPRS module configuration.

Returns
- UBX_S_PDP_CONFIGURED
- UBX_S_PDP_NOT_CONFIGURED

UBX_STATUS ubxGetPinStatus(void)

Description
This function returns the status of the PIN code.

Returns
- UBX_S_PIN_ENABLED
- UBX_S_PIN_NOT_ENABLED

UBX_STATUS ubxGetSimStatus(void)

Description
This function returns the status of the SIM card.

Returns
- UBX_S_SIM_PRESENT
- UBX_S_SIM_NOT_PRESENT
UBX_STATUS ubxGetSystemStatus(void)

Description
This function returns the status of the u-blox high-level interface.

Returns
• UBX_S_SYSTEM_CONFIGURED
• UBX_S_SYSTEM_NOT_CONFIGURED

Example
#include <plib.h>
#include "libubx.h"

UBX_ERROR rc;
rc = ubxConfigureSystem(SYSTEM_FREQ);
if (ubxGetSystemStatus() != UBX_S_SYSTEM_CONFIGURED)
{
   return;
}
DBPRINTF("\nSystem configured\n");

UBX_ERROR ubxPowerOffBoard(void)

Description
This function turns off power to the C16-G26Q plug-in board.

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

UBX_ERROR ubxPowerOffGps(void)

Description
This function turns off power to the GPS module.

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING
UBX_ERROR ubxPowerOnBoard(void)

Description
This function turns on power to the C16-G26Q plug-in board.

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

UBX_ERROR ubxPowerOnGps(UBX_GPS_ASSIST_MODE mode)

Description
This function turns on power to the GPS module.

Parameters
[in] mode GPS aiding mode

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

Example
See Example 3.

UBX_ERROR ubxRegisterGsmNetwork(void)

Description
This function registers to the GSM network.

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

Example
See Example 2.
UBX_ERROR ubxRegisterPdpService(void)

Description
This function registers to GPRS services.

Returns
- UBX_E_OK
- UBX_E_ERROR
- UBX_E_WARNING

Example
See Example 4.

UBX_ERROR ubxSendGsmShortMessage(char *gsmsn, char *sms)

Description
This function sends an SMS (text message) via the GSM network.

Parameters
[in]  gsmsn  GSM subscriber number
[in]  sms    Short message (160 character maximum)

Returns
- UBX_E_OK
- UBX_E_ERROR
- UBX_E_WARNING

UBX_ERROR ubxUnregisterGsmNetwork(void)

Description
This function unregisters from the GSM network.

Returns
- UBX_E_OK
- UBX_E_ERROR
- UBX_E_WARNING
UBX_ERROR ubxUnregisterPdpService(void)

Description
This function unregisters from the GPRS services.

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

UBX_ERROR ubxUpdateGpsContext(GPS_CONTEXT context)

Description
This function updates a GPS data context (read only selected message).

Parameters
[in] context GPS data context to update

Returns
• UBX_E_OK
• UBX_E_ERROR
• UBX_E_WARNING

Example
#include <stdlib.h>
#include <plib.h>
#include "libubx.h"
#include "libgps.h"

UBX_ERROR rc;
GPS_DATA_GGA *gga;

rc = ubxUpdateGpsContext(GPS_C_GGA);
gga = gpsGetGGA();

DBPRINTF("Altitude: %i m/sl\n", atoi(gga->alt));
DBPRINTF("Latitude: %f %s\n", atof(rmc->lat), rmc->lat_ns);
DBPRINTF("Longitude: %f %s\n", atof(rmc->lon), rmc->lon_ew);
**Description**

This function updates all GPS data context (read all messages).

**Returns**

- **UBX_E_OK**
- **UBX_E_ERROR**
- **UBX_E_WARNING**

**Example**

```c
#include <stdlib.h>
#include <plib.h>
#include "libubx.h"
#include "libgps.h"

UBX_ERROR rc;
GPS_DATA_GSV *gsv;
GPS_DATA_RMC *rmc;

rc = ubxUpdateGpsFullContext();
gsv = gpsGetGSV();
rmc = gpsGetRMC();
DBPRINTF("Number of satellites used for tracking: %d\n", atoi(gsv->nst));
DBPRINTF("Latitude: %f %s\n", atof(rmc->lat), rmc->lat_ns);
DBPRINTF("Longitude: %f %s\n", atof(rmc->lon), rmc->lon_ew);
```

---

**Description**

This function verifies the Pin code.

**Parameters**

| [in] | **pin** | Pin code (null-terminated string) |

**Returns**

- **UBX_E_OK**
- **UBX_E_ERROR**
- **UBX_E_WARNING**
BYTE acdEmailSetup()

Description
This function sets up initial communications with an SMTP server. The M2M Board does not have SSL capabilities. The sender of the e-mail is also setup here.

Returns
BYTE showing return value of SMTP request. Value is 0 if no error.

BYTE acdSendEmail(char *Recipient, char *Subject, char *Message)

Description
This function sends an e-mail.

Parameters
[in] Recipient character string of the e-mail to send the e-mail to
[in] Subject character string of the subject of the e-mail
[in] Message character string of the message of the e-mail

Returns
BYTE value showing return value of SMTP request, value is 0 if no error.

BYTE acdHTTPRequest(BYTE requesttype, char *Servername, char *FileLocation, char *Filename)

Description
This function sends an e-mail.

Parameters
[in] requesttype BYTE defining HTTP request type (HEAD, GET, DELETE, PUT, POST_FILE, POST_DATA)
[in] Servername Character string of server name
[in] FileLocation Character string of server file location
[in] Filename Character string of M2M file to save the file from HTTP server to LEON-G200 memory

Returns
BYTE value showing return value of HTTP request. Value is 0 if no error.
int acdReadM2MFile(char *Filename, unsigned char *File)

Description
This function sends an e-mail.

Parameters
[in] FileName Character string of Filename of local file
[in] File Character string of File in PIC memory

Returns
Integer value showing BYTE length of file read from M2M memory.
Stack API Usage Examples

EXAMPLE 1: INITIALIZING THE M2M PICtail™ PLUS DAUGHTER BOARD

ubxConfigureBoard();

EXAMPLE 2: INITIALIZING GSM

ubxConfigureGsm();

if (ubxGetSimStatus() == UBX_S_SIM_PRESENT) //Check to see SIM card is present
{
    ubxRegisterGsmNetwork();
    break;
}

if (ubxGetGsmNetworkStatus() == UBX_S_GSM_NETWORK_REGISTRED)
{
    acdSMSSetup(); //Setup SMS Reading
}

EXAMPLE 3: INITIALIZING GPS

ubxConfigureGps();
ubxPowerOnGps(UBX_GPS_ASSIST_OFFLINE);

EXAMPLE 4: INITIALIZING GPRS

if (ubxGetGsmNetworkStatus() != UBX_S_GSM_NETWORK_REGISTRED)
{
    ubxConfigurePdp(UBX_CONFIG_APN);
    break;
}

EXAMPLE 5: USING SMTP (SENDING AN E-MAIL)

acdEmailSetup(); //Sets up the SMTP server (generic is Yahoo server)
acdSendEmail(EmailAddress,"Microchip M2M Board Message","This is a message from Microchip's M2M Board.");

EXAMPLE 6: USING HTTP

acdHTTPRequest(GET, //Type of HTTP request
"maps.google.com", //HTTP server
temp, //file location on server
"Map"); //Read Google Maps JPEG

EXAMPLE 7: READING FROM MEMORY

"fileLength ="acdReadM2MFile("Map", &PIC32MapMemoryLocation[0]) //Reads file from M2M Board
//to PIC32 memory
GRAPHICS LIBRARY DESCRIPTION

The demonstration uses the Microchip Graphics Library, version 2.11, which is a powerful library that makes creating a Graphical User Interface (GUI) such as this one fast and easy. The Microchip Graphics Library is free and available for download from: www.microchip.com/MAL.

REFERENCES

LEON-G200 and NEO-6Q Communications Modules (www.u-blox.com)
LEON-G100 G200 “2G GPS/GPRS AT Commands Manual” GSM.G1-SW-09002 (www.u-blox.com)
M2M PICtail™ Plus Daughter Board (www.microchip.com)
Multimedia Expansion Board (MEB) (www.microchip.com/meb)
PIC32 device family (www.microchip.com/PIC32)
APPENDIX A: SOURCE CODE

Software License Agreement

The software supplied herewith by Microchip Technology Incorporated (the “Company”) is intended and supplied to you, the Company’s customer, for use solely and exclusively with products manufactured by the Company.

The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN “AS IS” CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

All of the software covered in this application note is available as a single WinZip archive file. This archive can be downloaded from the Microchip corporate Web site at:

www.microchip.com
APPENDIX B: SCHEMATICS

FIGURE B-1: LEON-G200 MODULE AND ANTENNA

[Diagram of Leon-G200 module and antenna connections, including various components such as VBAT, V_CHARGE, SPI_IRQn, etc., with labels for each pin and component position.]
FIGURE B-2:  POWER BOARD HEADER

J101

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

20 PIN 2.54STRIP MALE RIGHTANGLE

V_CHARGE  VBAT

CHARGESENSE

1-B3

V_GPS

C106 330μF 6.3V

C108 47pF 5%

U101  U102  U103  U104

CA05P4S14THSG

DTR  RTS  CTS  TxD  RxD  CTS  RxD  GPS_TxD  GPS_RxD  V_GPS_EN  20PIN  2.54STRIP  MALE  RIGHTANGLE  20PIN  2.54STRIP  MALE  RIGHTANGLE  20PIN  2.54STRIP  MALE  RIGHTANGLE  20PIN  2.54STRIP  MALE  RIGHTANGLE

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3

V_CHARGE  VBAT  CHARGESENSE  1-B3
FIGURE B-3: SIM CARD HOLDER

- **C101**: 47pF_5%
- **R108**: 0Ohm_5%
- **C107**: 0Ohm_5%
- **R106**: 0Ohm_5%
- **R105**: 0Ohm_5%
- **R104**: 0Ohm_5%
- **USB0002**: USB0002
- **D100**: SIM_RST
- **SIM_CLK**: SIM_CLK
- **SIM_VCC**: SIM_VCC
- **SIM_IO**: SIM_IO
- **SIM_RST**: SIM_RST
- **U100**: SLM76CF3201P VQFN-8-1
- **J100**: J100
FIGURE B-4: AMPLIFIER CIRCUIT

[Diagram of amplifier circuit with components labeled]

- **VBAT**
- **GND**
- **IC100**
- **R111** (330kΩ_5%)
- **R112** (10kΩ_5%)
- **R113** (22kΩ_5%)
- **R114** (820kΩ_5%)
- **R115** (10kΩ_5%)
- **R116** (0Ω_5%)
- **D102** (0603 ORANGE)
- **D103** (0603 RED)
- **C109** (100nF_10%)
- **NC7S202P5X**
- **GPIO1**

[Additional notes and connections as per the diagram]
FIGURE B-5: NEO-6Q GPS MODULE AND ANTENNA
FIGURE C-1: M2M BOARD LAYOUT (TOP ASSEMBLY)
FIGURE C-2: M2M BOARD LAYOUT (BOTTOM ASSEMBLY)
FIGURE C-3: GSM/GPS POWER SUPPLY DAUGHTER BOARD LAYOUT

Install J2 on bottom side of PCB

Route wires for MEB Power through the cutout holes starting from bottom side, going to top side and exiting the bottom side

U-Blox header soldered straight to board (J3)

PCB Front view
Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip’s Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.