MICROCHIP

AN3215

Serialized PRIME Service Node on ATSAM4CMx-DB

Summary

This application note shows how to add PRIME (PoweRline Intelligent Metering Evolution) communications to the meter demonstration boards ATSAM4CMx-DB. The metrology board works as a host controller connected to a PRIME board working as a modem. The communication between host and modem is established by means of a serial port. The Microchip Universal Serial Interface (USI) is used to serialize the components and services available in the Microchip implementation of the PRIME specification.

The goal of this application note is to explain how to integrate the metrology code of the ATSAM4CMx-DB boards with the PRIME, USI and Distribution Line Message Specification (DLMS) features, so that customers are able to adapt the code to their specific requirements. An explanation of the tasks to be performed by the Microcontroller Unit (MCU) is provided, analyzing priorities and critical actions. The application note includes sample code.

Table of Contents

1. Introduction

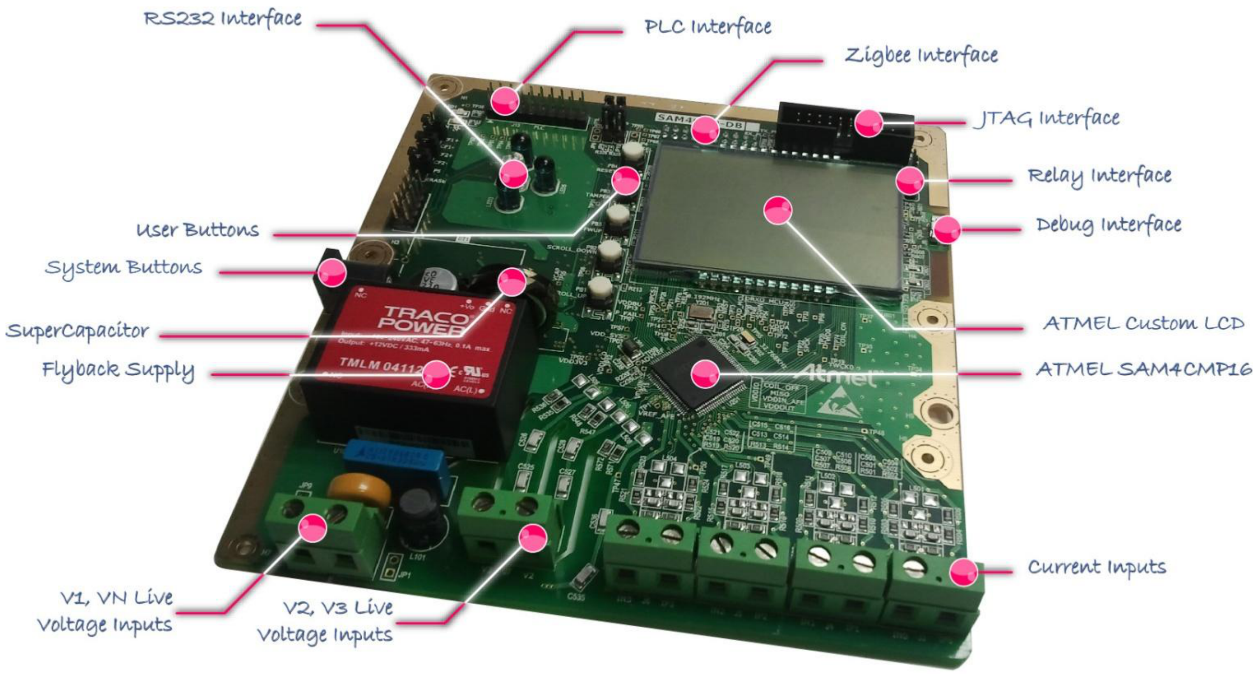
The ATSAM4CMS-DB and ATSAM4CMP-DB kits are meter demonstration boards for the 32-bit Arm® Cortex®-M4 SAM4CMx16C microcontrollers from Microchip:

- ATSAM4CMS-DB—for device SAM4CMS16C
- ATSAM4CMP-DB—for device SAM4CMP16C

Target demonstrations:

- Metrology performance verification
- Dual-Core Arm Cortex-M4 solution for meter firmware integration

Figure 1-1. ATSAM4CMx-DB Board



This application note shows how to add PRIME communications to the meter demonstration boards. The metrology board works as a host controller connected to a PRIME board employed as a modem. The communication between host and modem is established by means of a serial port. The Microchip Universal Serial Interface (USI) is used to serialize the components and services available in the Microchip implementation of the PRIME specification.

The target of this application note is to explain how to integrate the metrology code of the ATSAM4CMx-DB boards with the PRIME, USI and DMLS features, so that the customers are able to adapt the code to their specific requirements. Summarizing, these are the new features added to the meter firmware:

- Communications with the PRIME modem
- Registering of the device in a PRIME network
- Show how to retrieve metrology data using DLMS

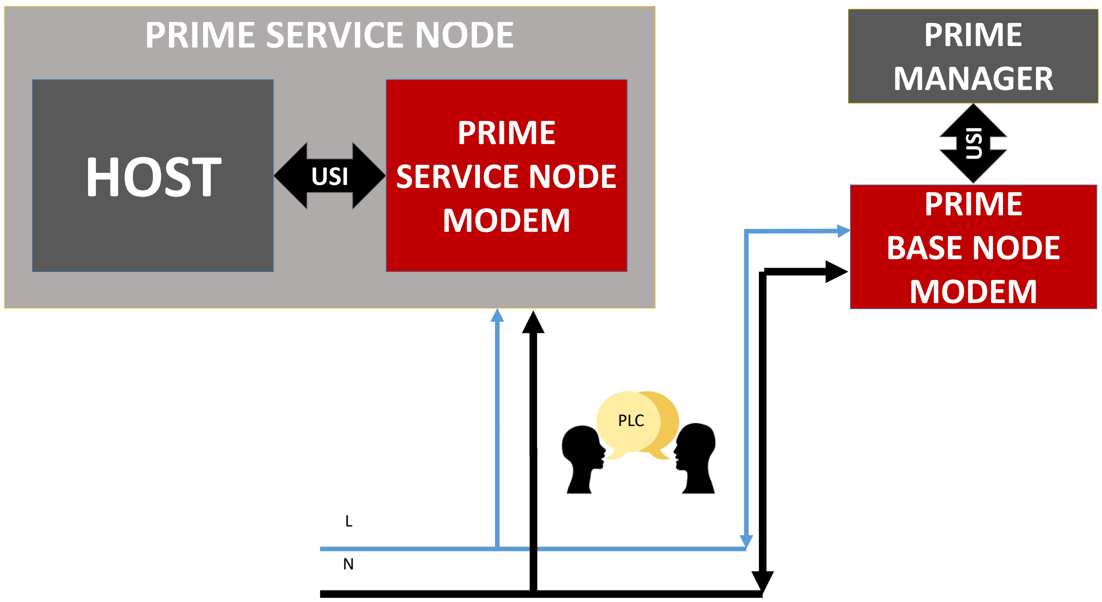
An explanation of the tasks to be performed by the MCU is provided, analyzing priorities and critical actions.

2. Hardware Requirements

2.1 Overview

The system is composed of a PRIME Service Node and a PRIME Base Node controlled by the Microchip PRIME Manager Tool. It is possible to add other devices to the PRIME network.

Figure 2-1. PRIME Network Schema



2.2 PRIME Service Node (Host + Modem)

The PRIME Service Node is composed of a host and a PRIME Service Node Modem. The host is implemented in the SAM4CMx MCU incorporated in the ATSAM4CMx-DB. There are several Microchip evaluation boards which can act as PRIME Service Node Modems, for example:

- ATPL230MB (ATPL230-EK)
- PL360MB (ATPL360-EK)
- SAM4CP16BMB (ATSAM4CP16B-EK)
- PL360G55CF-EK

The host and modem are connected by a serial port:

Figure 2-2. PRIME Service Node based on Modem Connection

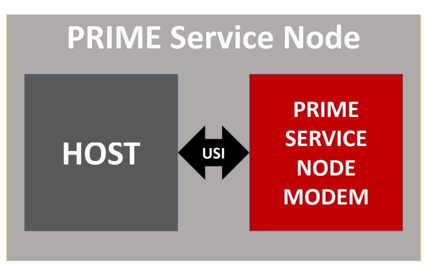


Table 2-1. Connection lines between Host and Modem

HOST	MODEM
TX	RX
RX	TX
GND	GND

2.3 PRIME Base Node

In order to establish a PRIME network, a PRIME Base Node is needed. There are several Microchip evaluation boards that can act as PRIME Base Node, for example:

- ATPL230MB (ATPL230-EK)
- PL360MB (ATPL360-EK)
- ATPL230ABN (ATBASENODE-EK)
- SAM4CP16BMB (ATSAM4CP16B-EK)
- PL360G55CF-EK

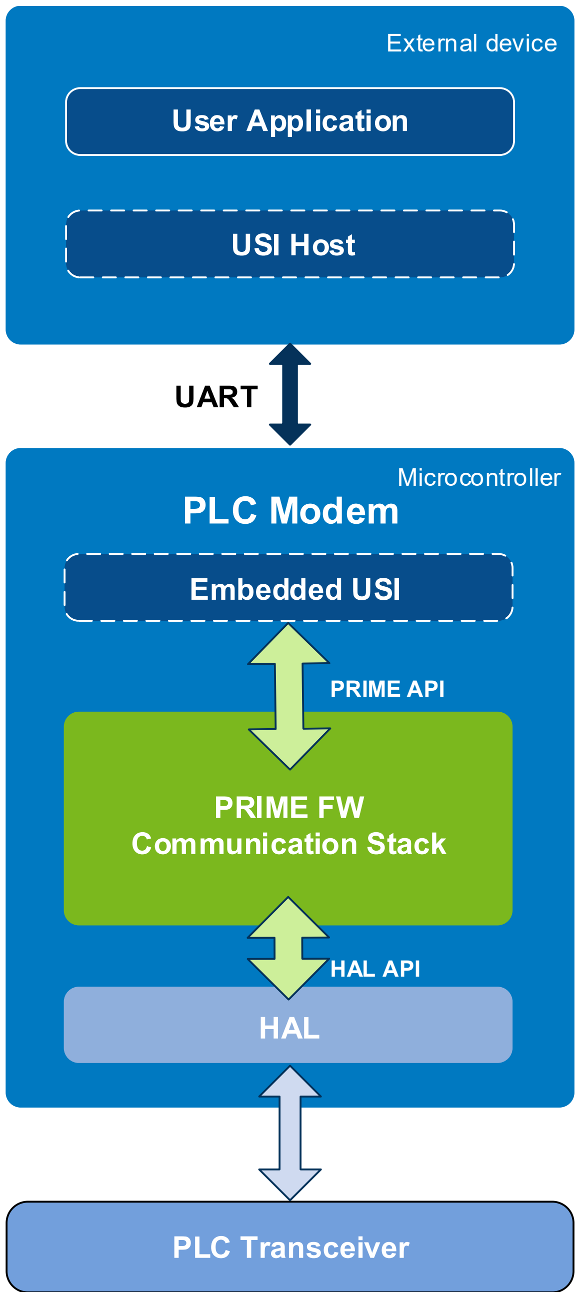
3. Firmware Loading

The service node modem and base node firmware are included in the standard PRIME firmware package developed by Microchip. Please program the boards with the firmware indicated in the following sections.

3.1 PRIME Service Node Modem

The PRIME Service Node Modem is an application example that brings access to the PRIME API through a serial connection. This application could be useful for users that want to make an intensive test of the PRIME stack or want to run the upper layers in another CPU. The interface with the PRIME Service Node Modem is USI. The overall architecture of this functionality is shown in the next figure.

Figure 3-1. Serialized PRIME Service Node Modem



In this architecture, files modem.h and modem_service.c in the embedded user application are in charge of coding and decoding the PRIME API primitives, whereas the Embedded USI (included in the HAL) is responsible for the transmission and reception of serial messages. Similarly, the USI Host in the external application is responsible for coding and decoding the PRIME API primitives and also for transmission and reception of serial messages.

The application allows control of the PRIME stack at different levels; the user can make use of the Convergence Layer API (CS) or directly use the MAC API as a shortcut for some tests.

The example is located on path *thirdparty\prime_ng\apps_1_4**prime_service_modem***. Please compile the project and program the board to be used as PRIME Service Node Modem.

3.2 PRIME Base Node Modem

The PRIME Base Node Modem connected to the PRIME Manager Tool is used to control the correct behavior of the application at the PRIME level. The PRIME Manager Tool allows management of a PRIME network and also includes a sniffer where traffic can be analyzed: MLME Register Indication, 4-32 Connection Establishment, DLMS data interchange, etc.

The PRIME Base Node Modem example is located on path *thirdparty\prime_ng\apps_1_4**prime_base_modem***. Please compile the project and program the board to be used as PRIME Base Node Modem.

Important: Microchip does not provide a PRIME Data Concentrator that requests DLMS objects from PRIME Service Nodes. Note that the Microchip PRIME Manager tool can act as a gateway placed between the Microchip BN and a PRIME Data Concentrator by means of DLMS over TCP.

3.3 Host

This is the firmware to be developed following this application note. The interface with the PRIME Service Node Modem is based on the "USI Host" package (more information available in document USI Host User Guide).

4. HOST Application Baseline Firmware

4.1 Overview

The host application is based on the ATSAM4CMx-DB firmware sample code. The objective is to show the customer how to add PRIME communications. The board works not only as a metering board, but also as a host controller capable of communicating with a PRIME modem. A DLMS example is provided, showing how to send metrology data by DLMS. Several key firmware packets have to be integrated:

- USI host
- PRIME management
- DLMS management

4.2 ATSAM4CMx-DB Application Firmware Sample Code

The starting point is the firmware developed for the ATSAM4CMx-DB, basically a meter demo application developed taking the Metrology Firmware 1.43 version as baseline. Please contact SmartMeterSupport@microchip.com to check for “Smart Meter Project” updates. The project files can be found in the following path: *SAM4CM_V1.43\Firmware\Application\Source Code*.

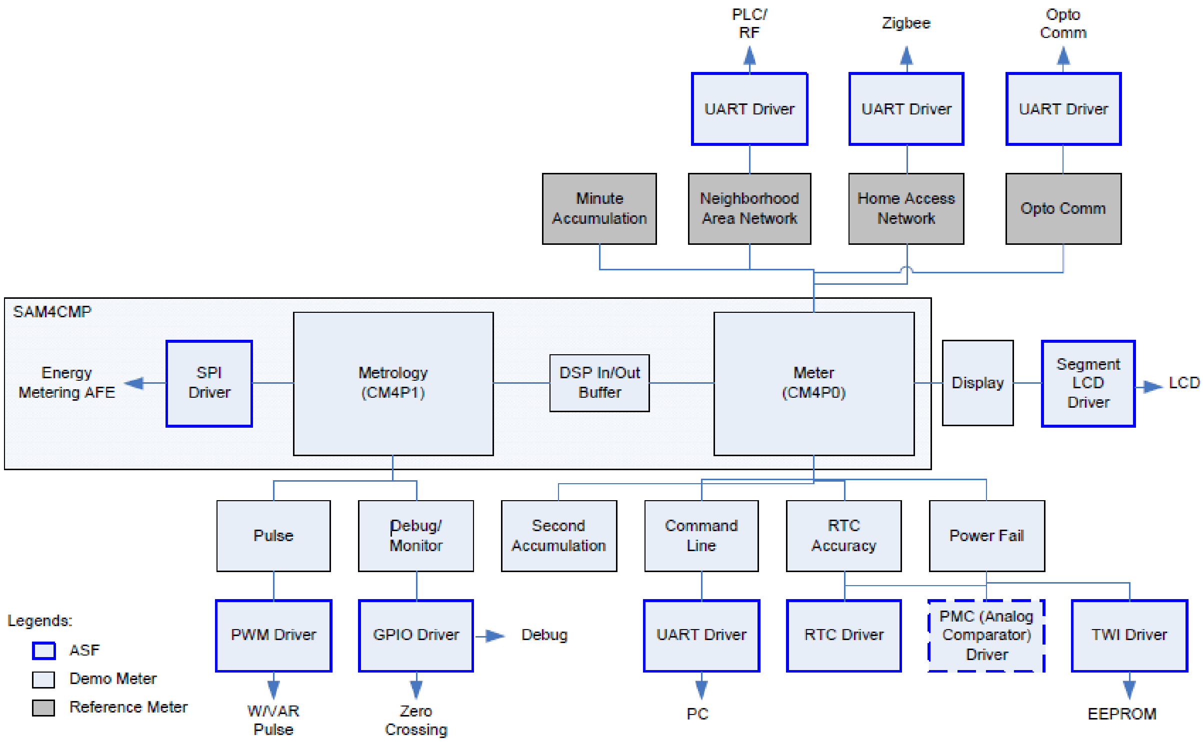
The “Smart Meter” package also contains the following documentation and help files:

Item	Location
Demo board hardware user guide	*SAM4CM_V1.43\Demo Board\SAM4CM HW user guide.pdf*
Demo board user guide	*SAM4CM_V1.43\Demo Board\SAM4CMx-Metering Demo User Guide.pdf*
Firmware metrology datasheet	*SAM4CM_V1.43\Firmware\Metrology\Metrology Datasheet\ Atmel_46005_SE_SAM4CMP16_Metrology_Datasheet_060514.pdf*
Firmware metrology user guide	*SAM4CM_V1.43\Firmware\Metrology\Metrology Datasheet\ Atmel_46204_SE_SAM4CMP16_Metrology_User_Guide_060514.pdf*
Firmware application developer guide	*SAM4CM_V1.43\Firmware\Application\Developer guide\SAM4CMx-Developer Guide.pdf*

The meter demo application implements a simple electric meter, in which Energy Metering AFE collects the data from current sensors and digitizes voltage and current data for further processing.

The meter demo application is an interrupt driven firmware with event flags to schedule different processing such as second processing, terminal user command processing, etc. The application runs on top of the Atmel Software Framework (ASF - device driver, services and components). Therefore, the low level firmware driver (ASF) handles the hardware setup and controls for both metrology firmware and meter firmware.

Figure 4-1. Meter Demo Firmware Block Diagram



In the metrology processor (CM4P1), metrology firmware collects the momentary interval raw data on every sample interval 16 kHz (62.5 μs). Metrology processor filters and decimates data to 4 kHz (0.25 ms) data rate for use in integration and calculation of all output data quantities updated at a rate by default set to 1 Hz.

The communications should not interfere with the data collection operation, which **is the critical task. The processor should collect the momentary data before the new data arrive**.

5. HOST Application Development

5.1 Step 1: Debug Tips

In order to simplify the debug process, some modifications have been included in the baseline code.

5.1.1 Default Dummy Handler

The default dummy handler has been modified (file *startup_sam4cm.c*) to have more information when an exception handler happens.

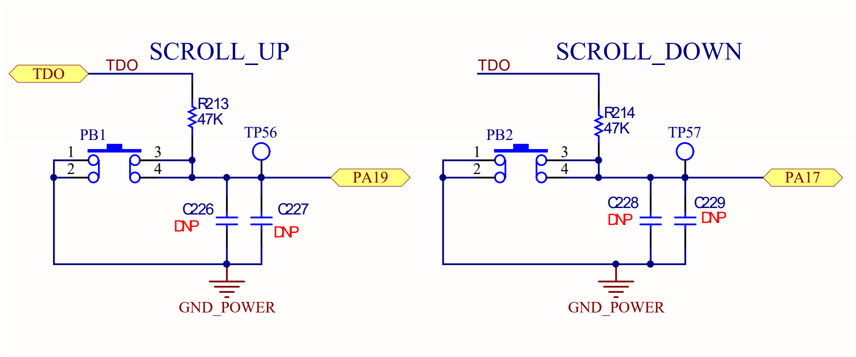
5.1.2 Watchdog Initialization

Watch debug halt (WDT_MR_WDDBGHLT) is activated, in order to stop it when the system halts in Debug state.

5.1.3 SCROLL UP and SCROLL DOWN Buttons

The TDO line is connected to SCROLL UP and DOWN buttons.

Figure 5-1. Scroll Down & Up Buttons



When debugging with JTAG, the TDO/RTC_1HZ/RTCOUT0 line activates the SCROLL_UP and SCROLL_DOWN buttons' interrupts. To avoid this unexpected activation, SCROLL_UP and DOWN interrupts have been disabled. This behavior is enabled by means of the pre-processor directive `#define SCROLL_DISABLED` (*conf_board.h*).

5.2 Step 2: Adding USI Serialization

5.2.1 Host and Modem Interconnection

The host communicates with the modem through the USI, so a free serial port is needed. The ATSAM4CMx-DB only has one available serial port, UART0, attached to PB4 and PB5.

The physical connection between the host and modem serial ports must be done in the following way, using a cable:

Table 5-1. Physical Connection lines between Host and Modem

HOST	MODEM
TX - PB5	RX
RX - PB4	TX
GND	GND

> **⚠WARNING** Please verify the modem configuration in order to know where to connect the modem wires. The settings are located in the *conf_usi.h* file of the `prime_service_modem` project: *thirdparty\prime_ng\apps_1_4**prime_service_modem***. Example selecting UART1 for USI:

```
#define USI_PORT_1      1
/* Port Communications configuration */
#define NUM_PORTS       1
#define PORT_1 CONF_PORT(UART_TYPE, USI_PORT_1, 230400, TX_UART_BUF0_SIZE,
RX_UART_BUF0_SIZE)
```

5.2.2 UART0 Multiplexing

UART0 is used by the terminal command interface. This interface provides access to the Microchip metrology library, meter parameters configuration, measurement results and line voltage status.

The meter demonstration board includes a multiplexer connected to UART0 and controlled by the microcontroller (PB15 GPIO). There are two possible connections:

- UART0 is redirected to the USB interface J9 (by default). This is the interface used by the terminal. It includes an isolator
- UART0 is redirected to the modem PLC interface J13

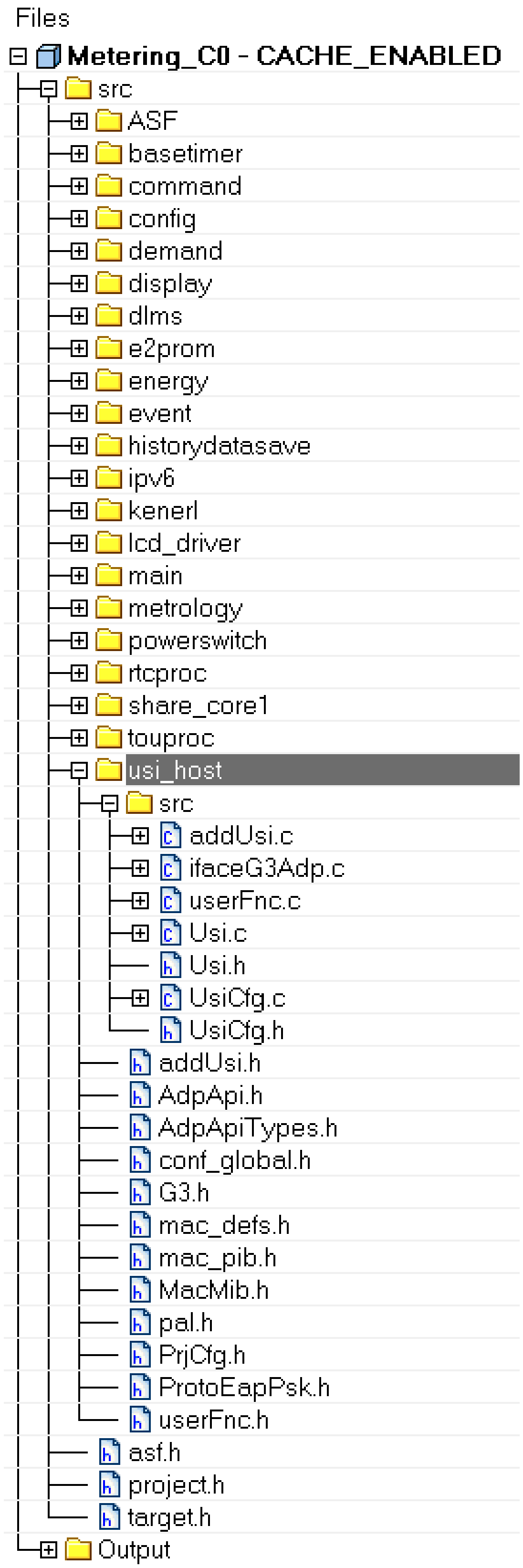
Therefore, the sample code includes a simple mechanism to select how the multiplexer is configured once the ATSAM4CMx-DB is switched on:

- The system connects UART0 by default to the modem PLC interface (USI connection)
- The system connects UART0 to the USB interface if there is a jumper joining pins 3 and 4 of J10 (relay interface)

5.2.3 Adding USI Files

The USI drivers are added to the project (usi_host folder).

Figure 5-2. USI Host files



Drivers to manage the UART0 and timers used by USI are included.

PrjCfg.h file contains the USI configuration. This file allows defining the ports to be used and their configuration. This is the configuration selected for this application:

```
#define NUM_PORTS                          1
#define PORT_0 CONF_PORT(UART_TYPE, 0, 115200, 512, 512)
#define NUM_PROTOCOLS                      1
//#define USE_MNGP_PRIME_PORT              0
//#define USE_PROTOCOL_SNIF_PRIME_PORT     0
#define USE_PROTOCOL_PRIME_API             0
//#define USE_PROTOCOL_ADP_G3_PORT         0
//#define USE_PROTOCOL_COORD_G3_PORT       0
//#define USE_PROTOCOL_SNIF_G3_PORT        0
//#define USE_PROTOCOL_MAC_G3_PORT         0
```

Please ensure that the baud rate of the modem is the same as the host one.

5.3 Step 3: Adding PRIME and DLMS Management

5.3.1 Task System Overview

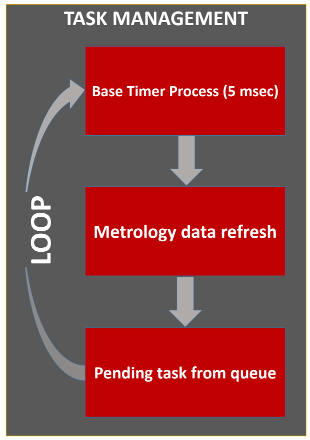
This section shows how to develop and integrate the DLMS and PRIME management tasks into the baseline standard metrology firmware of ATSAM4CMx-DB. The meter has three operating modes:

- Normal: The meter firmware measures V, I, energy and monitors line voltage input and meter hardware. It processes user commands and updates the Liquid Crystal Display (LCD) as required. The meter firmware transitions to Battery mode if power failed and battery voltage is higher than the minimum threshold
- Battery: The meter wakes up and the meter firmware turns on the LCD display every 6 seconds when the user pushes PB2 (configured as WKUP7 Input) or PB3 (configured as WKUP0 Input) buttons. Then, it turns off meter power (sleep). When power is supplied, it resets the MCU
- Restart: The meter firmware activates an external reset

In the provided example, the DLMS and PRIME management tasks are only executed in Normal mode, so they have to be integrated into the Normal mode task system. It is important to understand how the tasks are executed, so let's analyze briefly how the ATSAM4CMx-DB firmware manages the task system in Normal mode:

- Every 5 milliseconds the base timer process is executed. This process loads tasks periodically into the task queue
- New incoming metrology data from Core 1 is processed as soon as it is received. **This is a critical task, because data has to be processed before new data arrives**. The frequency of arrival of new data depends on user configuration
- If there are pending tasks in the task queue, one task is executed. Once executed, the next one will be processed after checking the base timer process event and the availability of new metrology data. The tasks loaded in the queue are related to Real Time Clock (RTC), Time Of Use (TOU), display, demand, serial communications, energy processing, calibration, history data save, harmonic analysis, etc.

Figure 5-3. Task Management



5.3.2 New Tasks Definition

Three new tasks are defined (*task.h* file):

- DlmsInit: Initialization and configuration of DLMS Server
- DlmsProc: DLMS management processing launched when the PRIME Connection 4-32 has been established
- UsiProc: USI processing

The new tasks are executed from the task queue.

USI processing is needed because PRIME management task communicates with the modem by means of USI. Base timer process loads USIProc task into the queue every 5 milliseconds. As this task is not critical, the frequency of execution can be lowered. The decision of lowering the frequency must be taken by the user, depending on the resources used by the application firmware.

The PRIME stack in Service Node automatically tries to register in the network. When this occurs, a register indication notifies the host through the USI interface, then the application tries to open a 4-32 connection to allow DLMS traffic be received/sent.

5.3.3 State Machines and Synchronous Communications

There are several PRIME functionalities implemented in the PRIME stack source code. In the modem, the application access to the data in the PRIME stack is very fast, as all the information is available in a single MCU. However, in the host-modem system, the user application is located in the host, while the PRIME stack is in the modem. Communication between them is done by means of USI, a serial interface, so accessing the information takes some time.

As the critical task is to collect the metrology data, it is not possible to wait for a long time to receive an answer from the modem. In other words, as the host manages critical processes, like the metrology data capture from Core 1, Core 0 cannot be blocked waiting for a response of the modem.

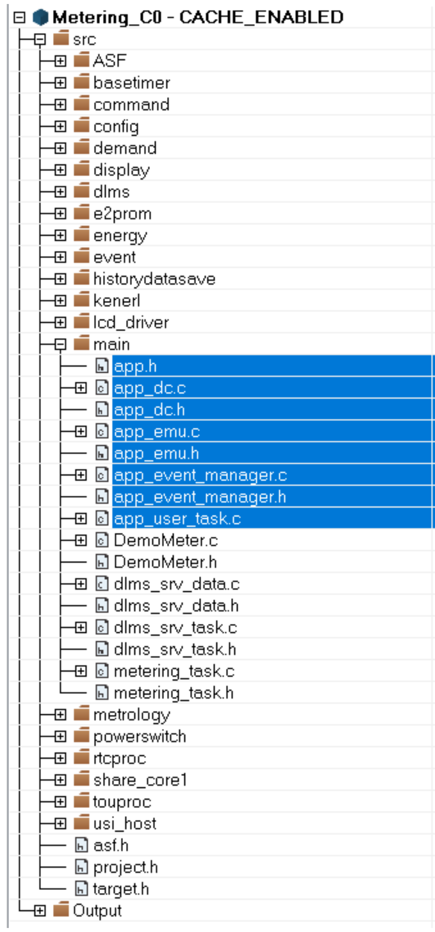
This situation happens when a synchronous communication is involved. To avoid blocking, the host application includes state machines which process the wait times by means of polling procedures, and additionally split the tasks in several sub-tasks.

5.4 Step 4: Adding PRIME

5.4.1 PRIME Application Files

The host application is based on `prime_service_dlms_met_app` project running on the mode defined by `DLMS_APP_MODE`. For modes `APP_EMU_MODE` and `ENABLE_DIRECT_CON_EXAMPLE_APP`, additional changes need to be made.

Figure 5-4. PRIME Application Files

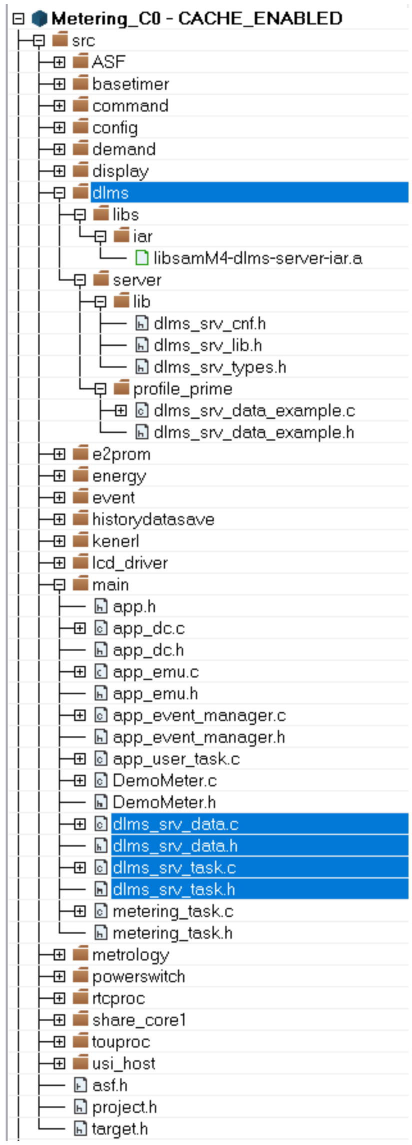


5.5 Step 5: Adding DLMS Lite

5.5.1 DLMS Application Files and Library

The ”dlms” folder contains the DLMS Lite files and library. Additionally, some files have been added to the “main” folder.

Figure 5-5. DLMS Lite files and library



The Microchip Embedded DLMS Lite libraries include basic functionality for PRIME stack testing. Please take into account that this library does not include a fully-featured DLMS implementation. More information about the DLMS library can be found in user guide *DS50002890 - Microchip Embedded DLMS Lite Libraries*.

The DLMS server library has two separate parts:

- A compiled library core to perform common DLMS operations: association management, headers processing, block reassembly, interface classes' common implementation, etc.
- The specific profile implementation module: *dlms_srv_data.c* and *dlms_srv_data.h*. This module is intended to be connected to the application responsible for processing all the information related to the collected data.

Header files contained in the *dlms/server/lib* folder are related to the library core and must not be modified by the user.

The user application is responsible for association configuration, library initialization and OBIS objects configuration.

The host application has its own dlms_srv_data files. These files contain the specific routine for each OBIS object. The request response payload is generated here. Within the library, OBIS code example functions implementation is provided.

By default, *dlms_srv_data.c* functions wrap *dlms_srv_data_example.c* functions present in the library. OBIS implementation examples allow basic DLMS testing. Example functions return dummy data similar to how a regular meter would.

The following sections show how to modify the project to link the metrology data with DLMS objects, to return measured data.

5.5.2 DLMS Initialization Process

The DLMS initialization task is implemented as a state machine. One of the states of DLMS initialization requires getting PRIME parameters from the modem: this is implemented with an additional state machine. The first time the “DlmsInit” task is loaded into the queue is at the meter initialization. Then, the system manages the reload mechanism. This mechanism is composed of two reload types:

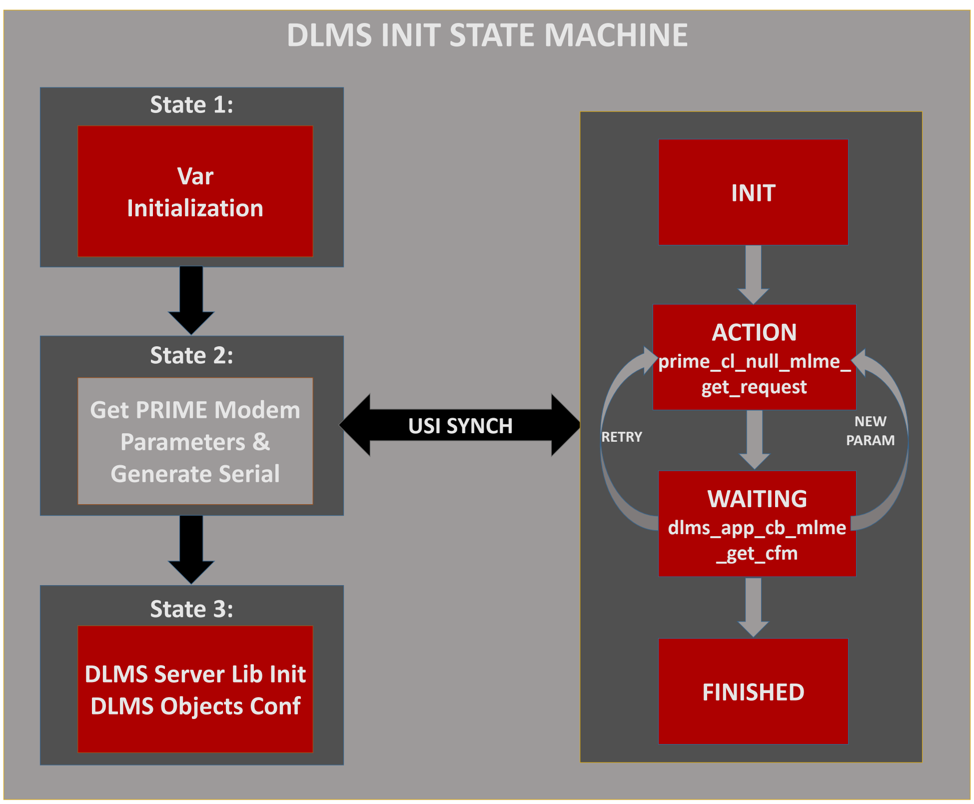
- Direct reload: The task loads itself into the queue. The reloading is accomplished by means of the “PutTaskIntoQue(DlmsInit)” statement.
- Indirect reload: Another process reloads “DlmsInit” task into the queue.

Direct reload is useful to fragment the number of actions to be performed consecutively. As the task queue executes only one task each time it runs, base timer and metrology processes can be executed more frequently.

Indirect reload is useful to ensure the task is reloaded with a timeout, or after executing other tasks. For example: when the host is waiting for the modem to answer, the reload is done by base timer process every 5 milliseconds. Additionally, before loading “DlmsInit” task, “USIproc” is loaded, because the answer to the host will not arrive until “USIproc” is executed.

This following diagram shows the initialization state machine implemented in the sample code.

Figure 5-6. DLMS Init State Machine



State 1 includes some variables initialization (related for example to 4-32 connection).

State 2 includes USI synchronous communication ”GetPrimeModemParameters” in order to get information necessary for correct DLMS Server Library initialization.

State 3 initializes DLMS Server Library and configures DLMS Server objects.

The parameters initialization procedure in state 2 is a good example to understand how to implement a synchronous USI communication. In this case a set of synchronous communications is performed, because each parameter requires a USI command to be initialized. The following states are considered:

- “Init”: initialization of the variable responsible for managing the number of times a communication is retried in case it fails
- “Action”: the USI command is launched. A timeout is defined
- “Waiting”:
  - If the answer has been received, the machine switches to “Action” state, in order to launch the next USI command. If there are no commands remaining, the machine goes to “Finished” state
  - If the answer has not been received and the timeout has expired, a retry is performed. If the number of retries has been finished the modem is reset
- “Finished”

The rest of USI communications are asynchronous in this example. The host will act against events coming from the Prime Service Node Modem running the corresponding callbacks:

- MLME Register Indication: received when the Prime Service Node is registered in the PRIME network. It will trigger a 4-32 connection establishment request (`prime_cl_432_establish_request`)
- MLME Unregister Indication: received when the SN is unregistered from the PRIME network. It will trigger the 4-32 connection release
- 4-32 Connection Establishment Confirm: received when a 4-32 connection is opened. It will trigger the “DlmsProc” task execution
- 4-32 Connection Release Confirm: received when a 4-32 connection is closed. It will trigger another 4-32 connection establishment request

“DlmsProc” is launched every 5 milliseconds only when the PRIME 4-32 connection is open.

5.5.3 DLMS Process

The DLMS process does not include synchronous communications and it does not require a high processing time, so a state machine is not needed. The processing does not have strict time requirements but the user application must ensure that it is called at least once every time a packet is received. Otherwise, a second packet would replace the previous one because reception is not buffered. Only one packet is processed every time DLMS process function is called.

5.5.4 DLMS Object Implementation Example

This chapter shows how to implement a DLMS object, allowing the application to retrieve data acquired from the metrology library.

As mentioned before, the DLMS default example code returns dummy data. This section shows how to implement the DLMS object 0-0:21.0.5.255.

This electricity related object is used to report instantaneous values of energy registers. The values to be returned are defined in the following table:

DLMS object 0-0:21.0.5.255			
Logical name	0000150005FF		

...........continued			
DLMS object 0-0:21.0.5.255			
Capture objects	8,0-0:1.0.0.255,2	Clock	For single-phase meters only one phase is active, so values connected with other phases (L2 and L3) have to be zero
	3,1-0:32.7.0.255,2	Voltage L1	
	3,1-0:31.7.0.255,2	Current L1	
	3,1-0:52.7.0.255,2	Voltage L2	
	3,1-0:51.7.0.255,2	Current L2	
	3,1-0:72.7.0.255,2	Voltage L3	
	3,1-0:71.7.0.255,2	Current L3	
	3,1-0:90.7.0.255,2	Current (sum over all three phases)	
	3,1-0:1.7.0.255,2	Active Power P+	
	3,1-0:2.7.0.255,2	Active Power P-	
	3,1-0:3.7.0.255,2	Reactive Power Q+	
	3,1-0:4.7.0.255,2	Reactive Power Q-	
	3,1-0:13.7.0.255,2	Power factor	

Step 1: Object configuration

The object is configured in the DLMS initialization phase with `void dlms_init (void)` procedure (file *dlms_srv_task.c* file) in the sample code:

```
// "Eletricity related objects --> instantaneous values --> Instanteneous values"
dlms_srv_conf_obis(0, 0, 21, 0, 5, 255, 7, obis_0_0_21_0_5_255_cb, &x_new_obis);
```

Step 2: Callback function modification

The callback function (`obis_0_0_21_0_5_255_cb` in this example) is located in *dlms_srv_data.c* file. This function is linked to `obis_0_0_21_0_5_255_example_cb`, placed in *dlms_srv_data_example.c*. As explained before, by default *dlms_srv_data.c* functions wrap *dlms_srv_data_example.c* functions present in the library. So now the default `obis_0_0_21_0_5_255_example_cb` function can be modified in order to include the customized code to return measured data instead of dummy values.

The sample code mixes dummy data (clock) and measured data. As it can be seen, instantaneous values returned are linked to `sx_meter_info` structure, which contains the data acquired from the metrology.

```
// Instantaneous Voltage L1
puc_resp_data[uc_resp_data_idx++] = DT_LONG_UNSIGNED;
puc_resp_data[uc_resp_data_idx++] = (sx_meter_info.Inst_V_rms_L1 >> 8) & 0xFF;
puc_resp_data[uc_resp_data_idx++] = sx_meter_info.Inst_V_rms_L1 & 0xFF;
```

6. Revision History

6.1 Rev A - 09/2019

Document	Initial document release.

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

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.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with

Trademarks

ISBN: 978-1-5224-5004-7

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820