Debugging an MQX application

Our customer was developing an application based on a Kinetis MCU using the Freescale MQX RTOS. When the application was ported on the specific client's board, it crashed before to reach the main and any task creation. This behavior revealed incoherency between the hardware and the software during the MQX initialization, where the code is not simple to understand and control. AC6 helped the customer to find and to fix the bug which was a hardware problem linked to the board power supply.

Starting a new Project on a LPC43xx MCU with FreeRTOS and stacks

Our customer was starting a new project based on an LPC43xx MCU (Cortex-M4 + Cortex-M0 cores) with FreeRTOS. Open source stacks were needed. AC6 supported the team to create communication between the two cores to unload the Cortex-M4. We also created an application to save and access data on the SD Card through the USB using the Fatfs filesystem. Moreover, AC6 also helped the customer to get started with the TCP/IP LwIP stack and to display information on a LCD using the EmWin GUI.

Porting a FreeRTOS project on a Renesas Core using an MPU

Our customer asked for an "RT3" training (Real Time Programming with FreeRTOS). It turned to an expertise because the target was based on a Renesas core. So the different labs were ported to the client board and the training adapted to the hardware. Especially, the FreeRTOS port for the Renesas core was detailed. Moreover, because this port did not support the MPU (Memory protection unit) which was needed, AC6 supported the client to add this functionality to the port and to develop a FreeRTOS MPU-based application.

Starting a project based on a EFM32 board using FreeRTOS

Our customer was starting a new project based on an EFM32 MCU (Cortex-M3). It was its first multi-task application. So after a training on how to use the FreeRTOS RTOS and to avoid the pitfalls of the Cortex-M3 port, AC6 helped the customer to design their software application architecture: How to divide the application in tasks and how to make the tasks communicate to create a reactive system limiting the shared resources? There was also a need for a very low consumption application, so a guidance to achieve this goal was providing implementing the Sleep modes coming with the EFM32 MCU.

Help for the maintenance of a critical embedded system

Our customer had to port an existing certified system on a new platform to manage obsolescence issues. The application used a version of VxWorks not supported on new processors and application migration to a newer version would have required a full certification of the entire software. We have helped our customer to port the required VxWorks version on their new platform as well as correct errors in some drivers.