Developing Embedded Solutions on Asymmetric Systems using QT

Fernando Luiz Cola
fernando.cola@emc-logic.com
Whoami?

Fernando Luiz Cola
Embedded Software Developer
Emc Logic

fernando.cola@emc-logic.com
Agenda / Objectives

1. Understand AMP and SMP architecture
2. Applications and solution where AMP is a good fit
3. Overview I.MX7 Processor
4. RPMSG and Inter Processor Communication
5. Bottom-up example application with Qt / Linux / FreeRTOS
Introduction AMP vs SMP
Symmetric Multi-Processing

- Single OS controlling two or more cores of the same architecture
- CPU shares memory space
- Dynamic Scheduling and load balancing
Asymmetric Multi-Processing

- Different OS on each core
- Different core architectures
- Each core may run full-feature OS, Real-time OS or baremetal code
- Inter process communication protocol
- Efficient when the application can be statically partitioned across cores
Example

Multi-Chip System

CPU 1

MPU

Non Critical Task

Linux

Critical Task

FreeRTOS

Same SoC AMP

Dedicated Channel
Internal Communication

External Communication
UART / I2C

Non Critical Task

Linux

Critical Task

FreeRTOS

Chip 1

MPU

Chip 2

Multi-Chip System
Applications Examples

Robotics / Real-Time
Applications Examples

Mobile / Sensor Acquisition
Applications Examples

Wearable / Low-Power
Overview I.MX7 Architecture
NXP I.MX7 Overview

- Dual Cortex A7 core + Cortex M4 core
- Master/Slave architecture
  - A7 is the master
  - M4 is the Slave
- Inter processor communication
- **MU** – Message Unit
- **RDC** – Resource Domain Controller
NXP I.MX7 Overview

RDC – Resource Domain Controller
NXP I.MX7 Overview

MU – Message Unit

- Enables two processors within the SoC to communicate and coordinate by passing messages (e.g., data, status, and control).
- Signal the other processor using interrupts.
Linux and FreeRTOS talking

**Linux - Master Domain**
- U-boot load and starts M4 Core and Linux Kernel
- RPMSG driver creates virtqueues and endpoints
- Notifies remote processor
- RPMSG driver waits for name service announcement
- Send/Receive messages

**FreeRTOS - Remote Domain**
- RPMSG app creates virtqueues
- Waits for link being up
- App creates endpoints and sends name service announcement
- Send/Receive messages
Application Development with Qt
Hybrid Linux Qt / FreeRTOS Demo

- IMU sensor (I2C) read by MCU
- Qt App read data from MCU using RPMSG
- Plot data on Linux using QtCharts
Hardware Setup

- Colibri iMX7D 512MB
- Iris Carrier Board
- 7” display
- MPU6050
Hardware Setup

Host PC

Uart 1

Iris Board + I.MX7

Master / Linux
(A7)

Remote / FreeRTOS
(M4)

Uart 2

I2C

Host PC

Iris Board + I.MX7

Master / Linux
(A7)

Remote / FreeRTOS
(M4)

I2C

MPU6050
Qt Development

- Toolchain (cross-compile, rootfs, libraries) generated by Yocto-Project

- Configure Qt Kit for i.MX7 using toolchain generated by Yocto

- QtQuick and QML on i.MX7 (no-GPU) Qt 2D Software Rendering

  - `qputenv("QMLSCENEDEVICE", QByteArray("softwarecontext"));`

- Chart Visualization via QtCharts

  - Add to your `.pro`: `QT += charts`

- QtCharts is GPLv3!
Architecture Overview

- Kernel space
  - /dev/RPMSG
  - rpmsg_char
  - Rpmg channel

- User space
  - Qt App

- Linux

- FreeRTOS
  - IMU Sensor
  - readI2CData
  - Read Data Task
  - Rpmg channel
class Realtime : public QObject
{
    Q_OBJECT
    Q_PROPERTY(int accX READ XAcc NOTIFY accChanged)
    Q_PROPERTY(int accY READ YAcc NOTIFY accChanged)
    Q_PROPERTY(int accZ READ ZAcc NOTIFY accChanged)
}

public:
    Realtime(QObject *parent = nullptr);
    virtual ~Realtime();

private:
    QFile rpmsgDevice;

signals:
    void accChanged();

public slots:
    void update();
};

Realtime {
    id: realtime
}

Main.qml
rpmsDevice.setFileName("/dev/ttyRPMSG");
rmsgDevice.open(QIODevice::ReadWrite);

qDebug() << "Get Sensor Realtime Data";
if(!rpmsgDevice.isOpen()){
    qDebug() << "RPMSG Device not open";
} else {
    int accx, accy, accz;
    QByteArray query("acc");
rmsgDevice.write(query);
rmsgDevice.flush();
    QbyteArray response = rpmsgDevice.readLine(64);
    sscanf(response.constData(),
        "x:%d,y:%d,z:%d", &accx, &accy, &accz);
}
Timer {
    id: timer
    property int index: 0
    running: true
    repeat: true
    interval: 1000
    onTriggered: {
        realtime.update();
        accx.append(index, realtime.accX);
        accy.append(index, realtime.accY);
        accz.append(index, realtime.accZ);
        index++;
        axisX.min++;
        axisX.max++;
    }
}
ChartView {
    id: chartview
    animationOptions: ChartView.NoAnimation
    theme: ChartView.ChartThemeDark
    antialiasing: true
    anchors.fill: parent

    ValueAxis {
        id: axisX
        min: -5
        max: 5
    }

    ValueAxis {
        id: axisY
        min: -10
        max: 10
    }
}

LineSeries {
    id: accx
    name: "accx"
    axisY: axisY
    axisX: axisX
}

LineSeries {
    id: accy
    name: "accy"
    axisY: axisY
    axisX: axisX
}

LineSeries {
    id: accz
    name: "accz"
    axisY: axisY
    axisX: axisX
}
Demo Communication between cores

https://www.youtube.com/watch?v=SnLAsSJPCBU

![Communication between cores](Image)

QtCon BR
Demo QT charts
References

- M4 Firmware - https://github.com/ferlzc/Asymmetric_QT_demo_firmware

- QT Application - https://github.com/ferlzc/Asymmetric_QT_demo
References

- Linux and Zephyr “talking” to each other in the same SoC
  

- OpenAMP Project Page - https://github.com/OpenAMP/

- An Introduction to Asymmetric Multiprocessing: When this Architecture can be a Game Changer and How to Survive It (ELC 2018)
  

- Asymmetric Multiprocessing and Embedded Linux (ELC 2017)
  
  https://elinux.org/images/3/3b/NOVAK_CERVENKA.pdf

- Toradex FreeRTOS on Cortex-M4 of Colibri IMX7
  
Fernando Luiz Cola
fernando.cola@emc-logic.com

https://www.linkedin.com/in/ferlzc/
https://www.emc-logic.com/