Table 1 - Multimedia Derived Requirements

Description	Derived From	MCAPI
Low footprint on constrained devices (code and data size)	Characteristic 1	Yes
Framework for heterogeneous data types	Characteristic 1	Yes
Framework for complex transport configuration, quality of service, message delivery semantics and queuing policies (FIFO, most recent only, etc)	Characteristic 2	Some, others can be built on top.
Minimal assumptions about operating environment, and services required	Characteristic 3	Yes
No unnecessary data movement if architecture has supporting features (e.g., shared memory)	Characteristic 7	Yes, zero copy functionality
Allow direct binding to hardware, allowing end-to-end message delivery rates in the order of 100 cycles	Characteristic 8 Characteristic 9 Characteristic 10 Characteristic 11	Yes, no MCAPI feature prevents it
Framework for word-by-word message send/receive	Characteristic 8 Characteristic 9 Characteristic 10 Characteristic 11 Characteristic 12	Yes
Allow blocking and non-blocking operations	Characteristic 3	Yes
Provide message matching framework, to allow selection of messages of particular type, priority, deadline, etc	Characteristic 18	No, can be built on top of MCAPI.
Allow multi-drop messages. E.g., potential for >1 senders to queuing point, >1 readers	Characteristic 16 Characteristic 18	No, can be built on top of MCAPI.

7.4.3 Pseudo-Code Example:

The pseudo-code scenario contains the execution of a DSP algorithm such as a FIR filter on a multicore processor system containing a general purpose processor and DSP. The high level steps are as follows:

1. Initialize communication channels between processors.
2. GPP sends data for processing to DSP.
3. GPP sends code to execute to DSP
4. GPP signals for the DSP to begin execution.
5. GPP retrieves processed data from DSP

```
/*
 * MCAPI 2.000 Multimedia Use Case
 * shared.h
 *
 */

#ifndef SHARED_H_
#define SHARED_H_

#define CHECK_STATUS(status)
/* Shared Header File, shared.h */
enum {
```