ANALOG DEVICES

AHEAD OF WHAT'S POSSIBLE™

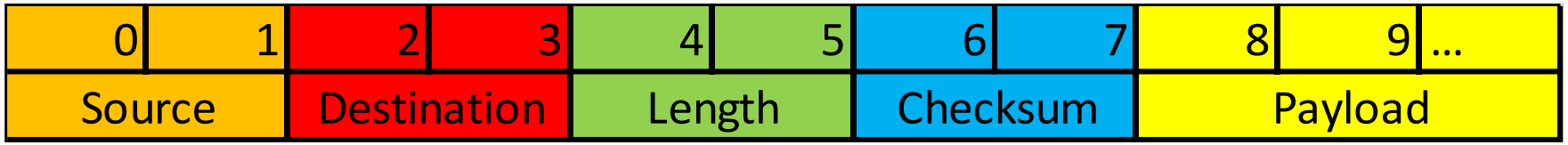
Machine to Machine Messaging V.2 (M2M2) Commands

RAJESH VIJAYAKUMAR

1/18/2021

M2M2 Header

- Based on variable-length UDP packets (2 bytes each for source, destination, checksum, length)
- Application-focused
  - All applications have an address
  - Packets are dynamically routed between applications - and across machines - based on addresses
- Implements Point-to-Point, Publish/Subscribe, and Broadcast messaging
  - A “Publisher” of data doesn’t have to know who is consuming that data, or where they are
- All inter-task messaging is done over M2M2
  - Individual tasks can be addressed transparently from anywhere in the system
  - Allows applications to be easily moved between machines in the system



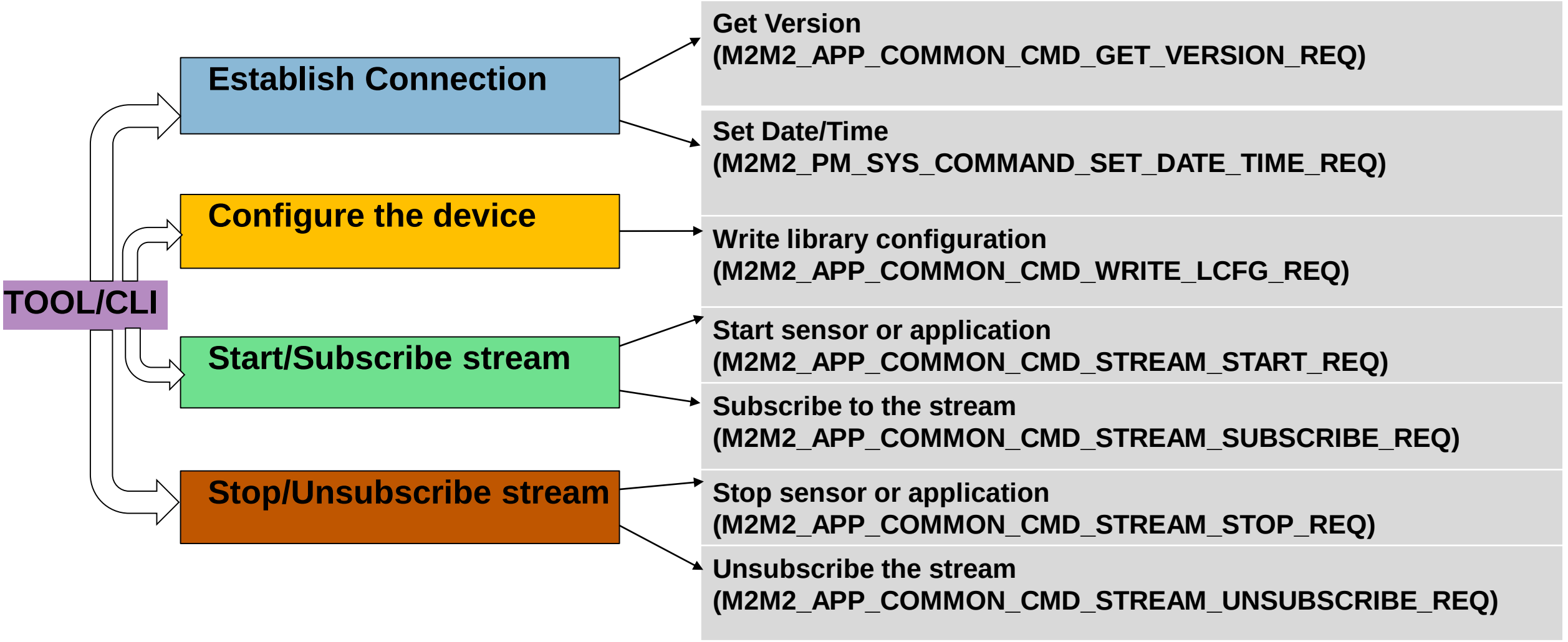
M2M2 – Commands Classification

- Each application or stream has an address - This is used to route packets across applications or machines
- Various sensor/bio-medical applications/streams supported
  - ADPD – ADPD stream
  - ADXL – ADXL stream
  - PPG – PPG stream;
  - ECG – ECG stream
  - EDA – EDA stream
  - Temperature – temperature stream
  - Pedometer – pedometer stream
  - Synchronized ADPD and ADXL – sync ADPD/ADXL stream
  - AD5940
- House-keeping/Utility applications/streams
  - File System
  - System application - Battery Stream information
  -

M2M2 – Getting Started with the sensor/bio-medical application

- Define the application and its stream address
  - M2M2_ADDR_MED_ECG = 49921 → Application
  - M2M2_ADDR_MED_ECG_STREAM = 50177 → Stream
- Define an interface file for the application - Interface contains all structs and enums used to interact with an application
  - Eg:- ecg_application_interface.py – single source of truth
    - The common application specific commands can be inherited from common_application_interface
    - The common sensor specific commands can be inherited from common_sensor_interface
  - Using data_definition_generator convert into language-specific definitions - C, C++ and python
  - Provides a single place to update the interfaces and have them propagate through the entire system (and tools)
- Application - Main components will be an Init(), message send() and process().
  - M2M2_APP_COMMON_CMD_STREAM_START_REQ → Init() → M2M2_APP_COMMON_CMD_STREAM_START_RESP
  - M2M2_APP_COMMON_CMD_STREAM_SUBSCRIBE_REQ → M2M2_APP_COMMON_CMD_STREAM_SUBSCRIBE_RESP
  - Process() – process data received from mailbox
  - Message_send() → All command responses
  - M2M2_APP_COMMON_CMD_STREAM_STOP_REQ → Deinit() → M2M2_APP_COMMON_CMD_STREAM_STOP_RESP
  - M2M2_APP_COMMON_CMD_STREAM_UNSUBSCRIBE_REQ → M2M2_APP_COMMON_CMD_STREAM_UNSUBSCRIBE_RESP

M2M2 – Getting Started with the sensor/bio-medical application…..



M2M2 – Command/Response Packets

M2M2_APP_COMMON_CMD_GET_VERSION_REQ

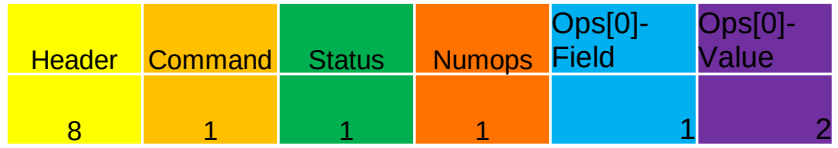
M2M2_APP_COMMON_CMD_GET_VERSION_RESP

M2M2_APP_COMMON_CMD_STREAM_START_REQ

M2M2_APP_COMMON_CMD_STREAM_START_RESP

M2M2_APP_COMMON_CMD_STREAM_SUBSCRIBE_REQ

M2M2_APP_COMMON_CMD_STREAM_SUBSCRIBE_RESP



M2M2_APP_COMMON_CMD_WRITE_LCFG_REQ

M2M2_APP_COMMON_CMD_WRITE_LCFG_RESP

M2M2 Tooling - Python Command Line Interface (CLI)

- Written in Python
- Just an M2M2 application
- Used as the reference tool for development
  - Firmware applications are developed and tested with the CLI first
- Easy to add new commands
- Easy to extend
- Easy to automate
- Built-in helptext

```
C:\Users\jzahn\Documents\git\gen3\m2m2\tools>CLI.py
This is the m2m2 UART shell. Type "help" or "?" to list commands.

#>?
NOTE: Not connected to a serial device!

Documented commands (type help <topic>):
========================================
batteryTest       fs_vol_info              getVersion    reg
clockCalibration  getAdpdVersion           help          sensor
connect           getAdxlVersion           lcfgEcgRead   setBatteryCharging
exit              getBatteryInfo           lcfgEcgWrite  setBatteryThreshold
flush             getDateTime              lcfgPpgRead   setDateTime
fs_format         getDcfg                  lcfgPpgWrite  setDateTimePS
fs_log            getEdaVersion            loadAdpdCfg   setLed
fs_ls             getLcfg                  loadAdxlCfg   setPowerMode
fs_mount          getLed                   loadDevice    setPowerModePS
fs_refhr          getPpgAlgoVendorVersion  msg_verbose   setPpgLcfg
fs_rm             getPpgStateInfo          ping          setPpgSync
fs_status         getPpgStates             plot          setSlot
fs_stream         getPpgSync               quickstart    status
fs_sub            getPpgVersion            quickstop     sub
fs_sub_status     getSystemInfo            raw_msg       usbPwr

Undocumented commands:
======================
disco

#>
```

```
#>?status
NOTE: Not connected to a serial device!
Get status of a sensor or application.

Available devices:
'ecg': The ECG service.
'eda': The EDA service.
'ppg': The PPG heart rate service.
'adxl': The ADXL device.
'syncppg': The sync PPG data stream service.
'adpd': The ADPD device.
---------------------------------------------
Usage:
    #>status [device]

    #>status adpd
    #>status adxl
    #>status ppg

#>?usbPwr
NOTE: Not connected to a serial device!

Set/Get the status of the ADP5350's DCDC converter.
This has the effect of enabling or disabling USB bus power to the device.
-------------------------------------------------------------------------------
Usage:
    #>usbPwr [get/enable/disable]
    #>usbPwr enable

#>
```

M2M2 Tooling - Basic CLI Commands for an application

- getVersion
  - Gets Version of PM and PS
- fs_log_start/stop
  - Start or stop logging
- fs_stream <file>
  - Reads a file from the NAND flash
  - Tracks file transfer rate and progress
- quickstart ecg
  - Run a pre-defined set of CLI commands with one shortcut
- msg_verbose
  - Changes the CLI's verbosity level
  - Can dump raw hex values of packets
  - Coloured messages to distinguish between verbosity levels