EgyRev SAD

IEEE 802.15.4-2006 Implementation SAD

This document describes the SAD (System Architecture Document) of the EgyRev project which is an implementation of IEEE 802.15.4-2006 standard

2011

Hesham Sh. El-Helw

Open Source

9/5/2011

# Purpose

This document describes the system architecture of EgyRev project which is an **OPEN SOURCE** implementation of IEEE 802.15.4-2006 standard. This document aims to provide the detailed description of the implementation to the level of the component design details.

This document is intended to be used by:

* System architect
* Component developer
* Component tester (white-box testing); if he is different from the component developer

However it can be viewed by anyone interested.

# Introduction

As an important step before jumping into coding and debugging, it’s very important to have a guidance of a good design which decreases system implementation (coding) task complexity and safe huge time in fixing some bugs that were be able to be avoided with good design in addition to fraction the project life time into steps.

This document is divided mainly in two sections, an overview section which contains the system building blocks (software modules) description followed by detailed components design description.

ICM handlers naming conventions:

* <DestCompName>\_ICM\_<MagName>\_<MsgType>
* DestCompName: The name of the component handling the ICM (MCM/MDM/.. etc)
* MsgName: An appropriate name for the message
* MsgType: ‘req’ for request, ‘’ for confirmation, ‘rsp’ for response, ‘ind’ for indication, ‘cnf’ for confirmation

# Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rev | Date | Editor | Section(s) | Description |
| 0.01 | 05/09/2011 | Hesham | All | Initial draft |
|  |  |  |  |  |
|  |  |  |  |  |

# Table of contents

[1. Purpose 2](#_Toc303634664)

[2. Introduction 2](#_Toc303634665)

[3. Revision History 2](#_Toc303634666)

[4. Table of contents 2](#_Toc303634667)

[5. Abbreviations 3](#_Toc303634668)

[6. Overview 3](#_Toc303634669)

[7. Detailed description 3](#_Toc303634670)

[7.1. MAC control manager (MCM) 3](#_Toc303634671)

[7.1.1. MAC reset 3](#_Toc303634672)

[7.1.2. Passive scan 3](#_Toc303634673)

[7.1.3. Association 4](#_Toc303634674)

[7.1.4. Disassociation by higher layer 5](#_Toc303634675)

[7.2. MAC data manager (MDM) 5](#_Toc303634676)

[7.3. MAC security manager (MSM) 5](#_Toc303634677)

# Abbreviations

MCM MAC Control Manager

MDM MAC Data Manager

MSM MAC Security Manager

MDB MAC Data Base

ICM Inter-Component Message

# Overview



Figure : System architecture

# Detailed description

## MAC control manager (MCM)

### MAC reset

* Initialize MAC\_DSN (Data Sequence Number) to a random value

### Passive scan

MCM receives the command MCM\_ICM\_Scan\_Req(ScanType = 0x02) upon which MCM do:

* Store current PAN ID that device is associated with in MCM private info using MDB\_GET\_PAN\_ID() to get its value from MDB.
* Set current PAN ID to 0xFFFF using MDB\_SET\_PAN\_ID(0xFFFF) to enable receiver to accept all beacons from all PAN coordinators not only the PAN coordinator the device is associated with.
* Request from MDM to discard any data packets except for beacons by triggering ICM\_SET\_FILTER\_TYPE(filterType = ONLY\_BEACONS)
* Set device state to STATE\_PASSIVE\_SCAN using MDB\_SET\_STATE(STATE\_PASSIVE\_SCAN).
* Trigger ICM\_SET\_REQ(PIBAttribute = phyCurrentPage, PIBAttributeValue = received value in MCM\_Scan\_Req)
* For each channel in the bitmap received in MCM\_Scan\_Req():
  + Trigger ICM\_SET\_REQ(PIBAttribute = phyCurrentChannel, PIBAttributeValue = received value in MCM\_Scan\_Req)
  + Trigger ICM \_SET\_TRX\_STATE\_REQ(state = RX\_ON) to enable the receiver
  + Start storing the received beacons till CFG\_MAX\_NUMBER\_OF\_BEACONS\_PER\_CHANNEL is reached.
  + Go to the next channel till the last channel in the bitmap.
* Upon completion of all requested channels, ICM\_SCAN\_CONFIRM is triggered to indicate the completion of the procedure.

MCM receives the interface MCM\_ICM\_Set\_Cnf(status) upon which it do:

* Nothing

MCM receives the interface MCM\_ICM\_SetTRX\_State\_Cnf(status) upon which it do:

* Nothing

MCM receives the interface MCM\_ICM\_Data\_Ind() upon which it do:

* Check the current state, if it’s STATE\_PASSIVE\_SCAN
  + Check the message type, if it’s beacon
    - Check whether macAutoRequest flag is set to FALSE, if yes
      * Trigger ICM\_BEACON\_INDICATION(recorded beacon info)
    - Else
      * TBD
  + Else
    - Error
* Else
  + TBD

### Association

MCM receives MCM\_Associate\_req() upon which it do:

* Set current PAN ID to the value received from HAL using MDB\_SET\_PAN\_ID(0xFFFF).
* Check whether the PAN coordinator address type in the request from HAL is set to be short or extended, if short
  + Set the short PAN coordinator address in MDB to the received value from HAL using MDB\_SET\_COORD\_SHORT\_ADDRESS(received value from HAL)
* Else
  + TBD
* Trigger ICM\_SET\_REQ(PIBAttribute = phyCurrentPage, PIBAttributeValue = received value in MCM\_Scan\_Req)
* Trigger ICM\_SET\_REQ(PIBAttribute = phyCurrentChannel, PIBAttributeValue = received value in MCM\_Scan\_Req)
* Compose the association request command air message.
* Trigger ICM\_SEND\_AIR\_MESSAGE() to send the association request command to the coordinator.

Upon receiving ICM\_AIR\_MESSAGE\_SENT(), MCM do:

* Start MAC\_WAIT\_TIME timer

Upon timer expiration, MCM do:

* MCM triggers MDM to ask for data pending for the device at the coordinator side by triggering ICM\_RECEIVE\_DATA()
* Restart MAC\_WAIT\_TIME timer

Upon receiving association confirm command from coordinator MCM do:

* checks the association status field, if it’s success:
  + triggers ICM\_ASSOCIATE\_RESPONSE()
  + Store the macShortAddress using MDB\_SET\_SHORT\_ADDRESS()
  + Store the coordinator extended address using MDB\_SET\_COORD\_EXTENDED\_ADDRESS()
* Else
  + TBD

### Disassociation by higher layer

Upon receiving MCM\_Disassociate\_req() MCM do:

* Check the current mode, if it’s associated
  + MCM compose the disassociation notification command
  + Trigger ICM\_TRANSMIT\_DATA() to send the dis-association notification command.
  + Reset the MAC attributes related to PAN to the default values (macPANID, macShortAddress, macAssociatedPANCoord, macCoordShortAddress, macCoordExtendedAddress)
* Else
  + ERROR

## MAC data manager (MDM)

### Data transmission

### Unslotted CSMA data transmission

Upon reception of MDM\_REQ\_TransmitData(), it do:

* Insert MAC\_DSN in the sequence number field of the data command and increment the stored MAC\_DSN with one
* Check the received source address mode in the command received from HAL, if it’s set to extended address (0x03)
  + Set the source addressing mode in the frame control field of the data frame to extended address (0x03)
  + Set the source address field to the stored extended address
* Else,
  + TBD
* Check the destination address mode in the command received from HAL, if it’s set to extended address (0x03)
  + Set the destination addressing mode in the frame control field of the data frame to extended address (0x03)
  + Set the source address field to the stored extended address
* Else,
  + TBD
* Checks if the destination PAN ID is the same as the current PAN ID, if yes:
  + Set the PAN ID compression bit in the frame control of the data command to 1
  + Clear the source PAN ID address bits in the frame control (0x00)
* Transmit packet with un-slotted CSMA-CA

***NOTE***: Only non-beaconing PAN’s are supported in R1.0

### CSMA-CA

Upon new data transmission request using un-slotted CSMA-CA, MDM shall do:

* Reset NB (number of back off’s counter) to zero
* Reset BE (Back off exponent) to minBE value
* Set MDM state to TRANSMITTING\_DATA
* Start timer CSMA\_CA\_BAackoffTimer with value (2BE - 1) backoff periods
* Request from PHY to perform CSMA
* Upon detection channel idle, request from PHY data transmission

### Data reception

## MAC security manager (MSM)

TBD