EVLA MIB Software Critical Design Review

Pete Whiteis, Software Engineer

MIB Software

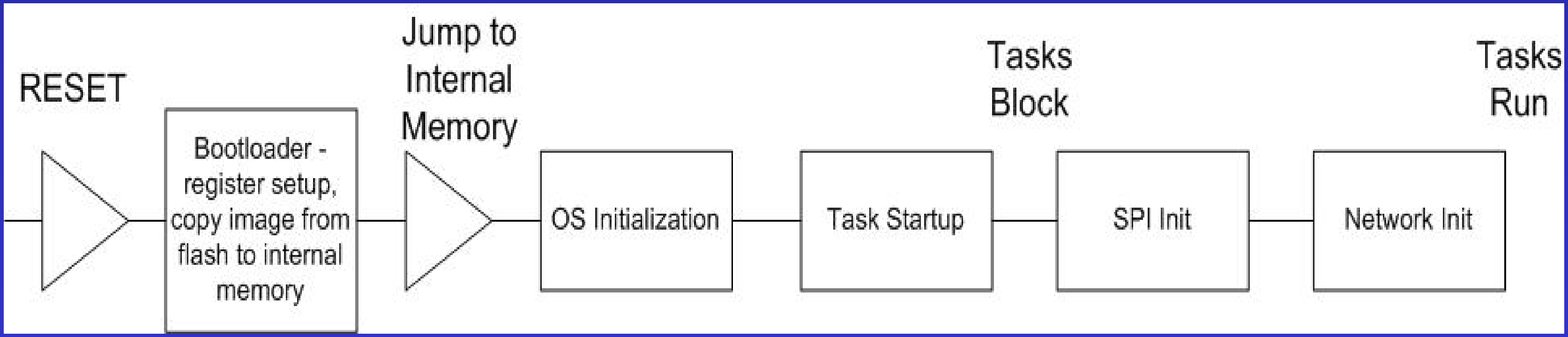
- Overview
  - – Systems Software (RTOS, Network Stack)
  - – MIB Framework .. Generic for all MIB’s
  - – Module Specific SW
  - – Rationale for MIB Framework
    - Rapid Code Development
    - Abstract HW differences using data driven design
    - Maximize Code Reuse – 95%

MIB Framework Software

- Overview
  - Framework Requirements
    - SPI for module communication
    - Ethernet for external communication
    - Use of internal memory
    - Software upgrade capability

MIB Framework Software

- Initialization



MIB Framework Software

- Memory Use
  - EDRAM: 512K for program code
  - ComDram: 1024K data storage
  - Flash Memory: Code and Data storage

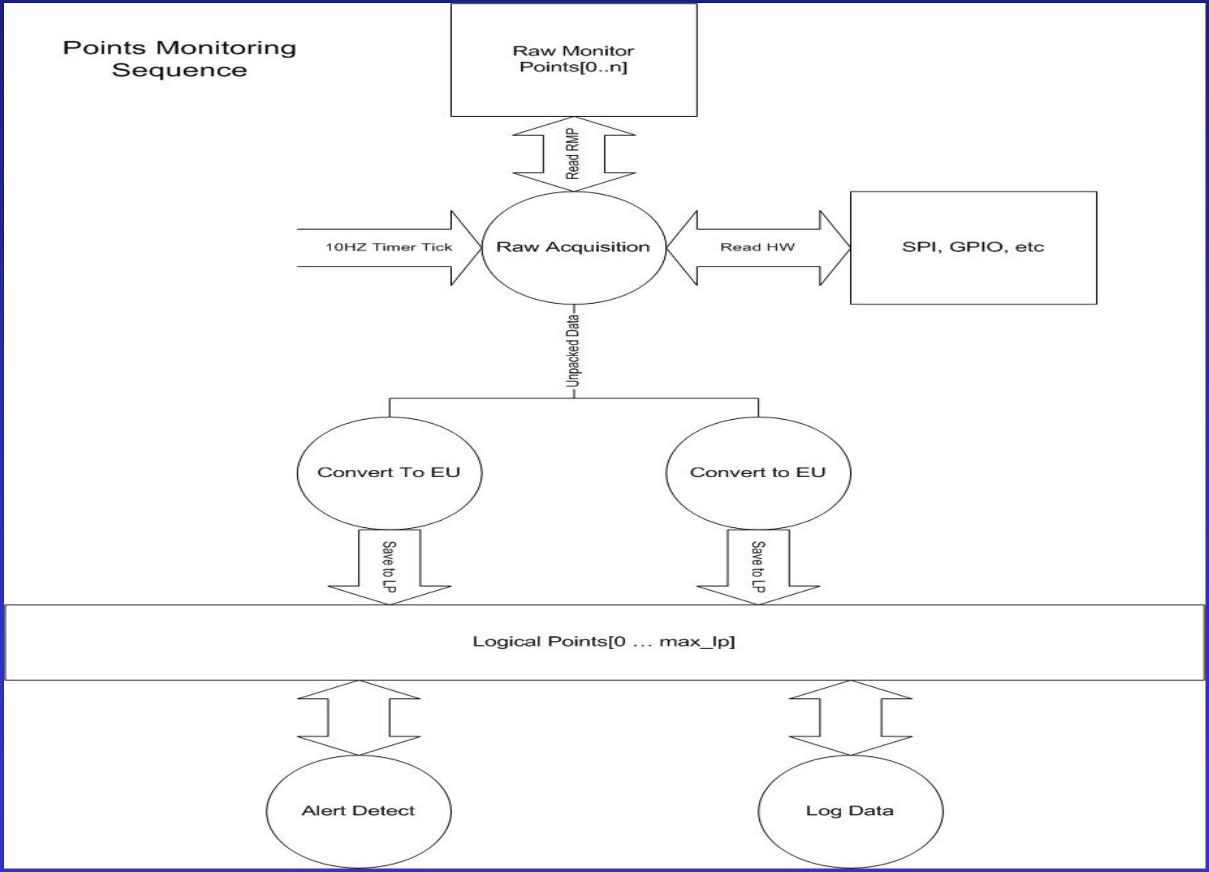
MIB Framework Software

- Data Structures
  - Logical Points
    - Global, memory resident DB
    - Array of structures which characterize an I/O point for a device
    - Monitor or Control, Analog vs Digital
    - Defines Alert criteria, conversion type, logging intervals
    - Accessible through command line
    - Initialized from XML Flash File

MIB Framework Software

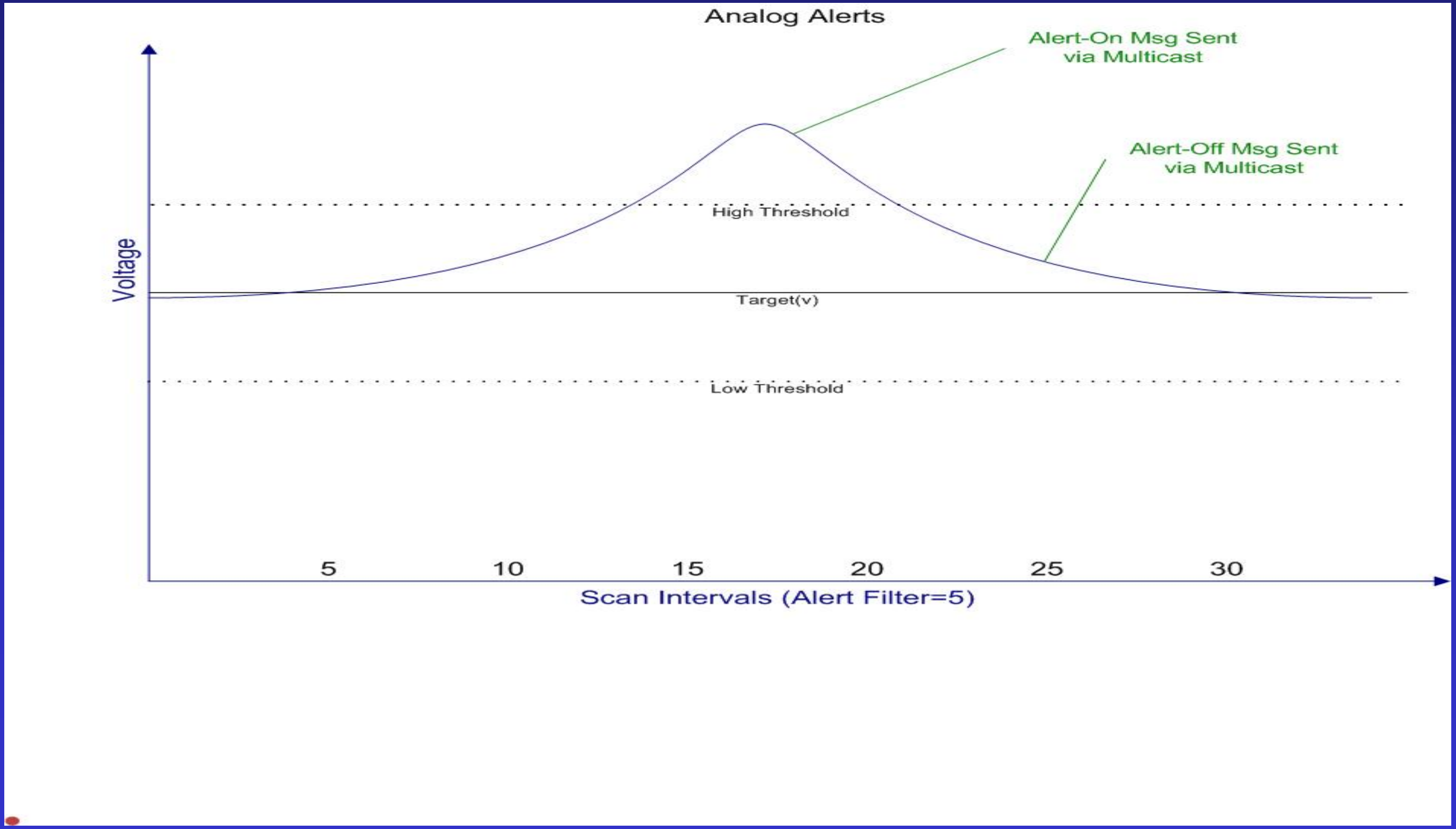
- Data Structures
  - Raw Monitor Points
    - Contains HW specific details
    - Describes origin and destination of RAW data
    - defined in module specific file ptsmon_usr_init.c

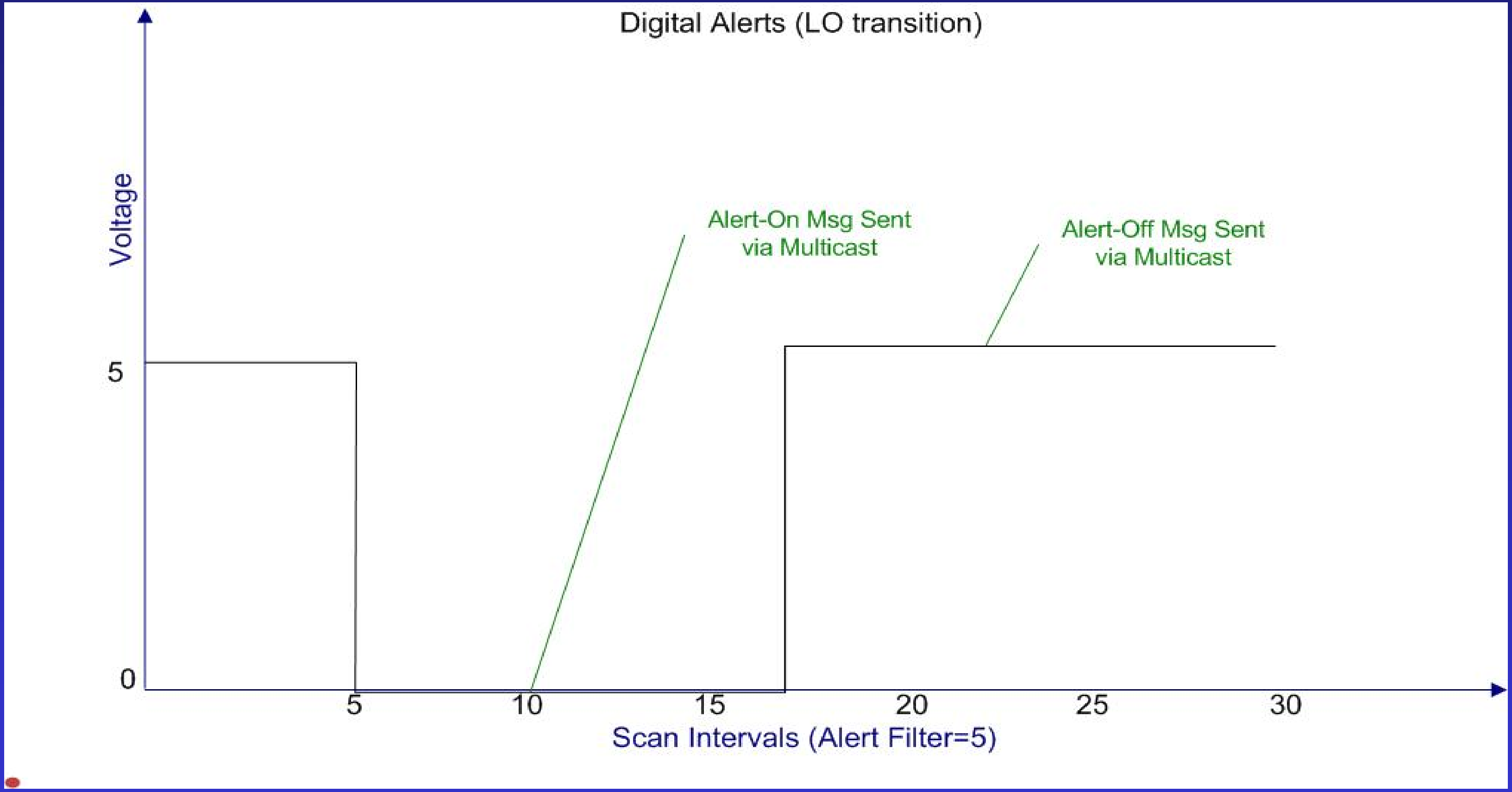
MIB Framework Software



MIB Framework Software

- Points Monitoring (cont)
  - All attributes used in conversions, alert detection and data logging rates are modifiable during runtime


Analog Alerts
Alert-On Msg Sent via Multicast
Alert-Off Msg Sent via Multicast
High Threshold
Target(v)
Low Threshold
Voltage
5
10
15
20
25
30
Scan Intervals (Alert Filter=5)


Digital Alerts (LO transition)
Voltage
5
0
Alert-On Msg Sent via Multicast
Alert-Off Msg Sent via Multicast
5
10
15
20
25
30
Scan Intervals (Alert Filter=5)

MIB Framework Software

- Wall Time
  - Used for data logging, deferred commands
  - NTP
  - 19.2Hz interrupt

MIB Framework Software

- High Resolution Timer
  - Used when OS timer inadequate
  - Repeating 50uSec minimum
  - Used like Sleep or semaphore

MIB Framework Software

- Field Upgradable Software
  - – Live upgrades of system image or points configuration
  - – Written to minimize data transfer time, maximize reliability

MIB Framework Software

- Commands
  - Simple ASCII interface
  - via TCP (Telnet) or UDP
  - ‘get’ or ‘set’
  - Use of XML output
  - time deferred (queued)

Module Specific Software

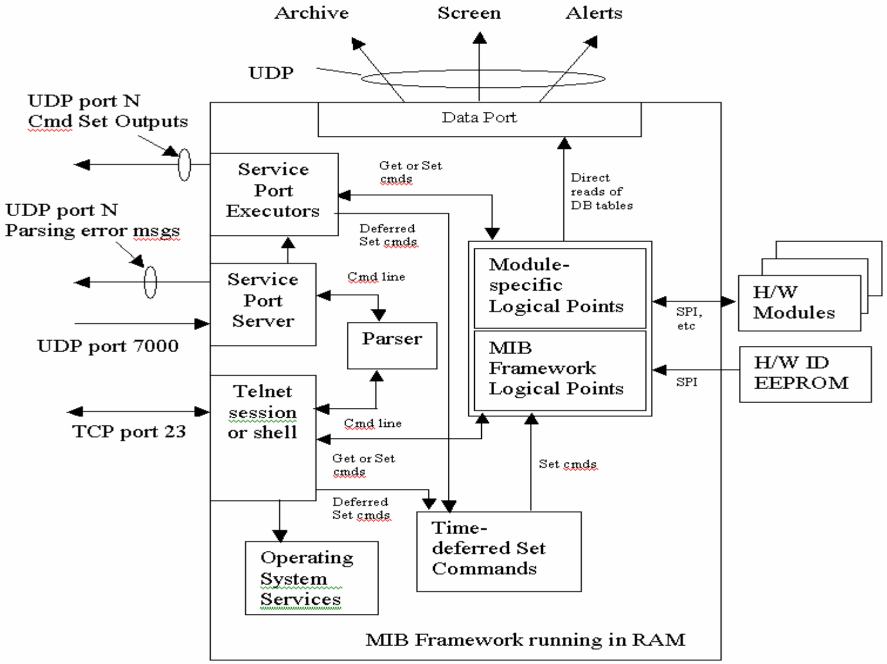
- Demo: Cmd I/F and file load

```
<Logical_Pts>
    <device name="L302">
        <monitor name="AGCV" type="analog" a_period="600" s_period="50"
                    o_period="600" aa_period="50" target="0" conv_type="LINEAR”
                    slope=".001" intercept="0" />
        <monitor name="PWR" type="analog" a_period="600" s_period="50“
                    o_period="600" aa_period="50" target="1.9" conv_type="LINEAR”
                    slope=".001" intercept="0" />
        <monitor name="LOCK2" type="digital" a_period="600" s_period="50“
                    o_period="600" aa_period="50" alert_arm="1" alert_on1="0" />
        <control name="YIG2_DAC" type="analog" a_period="600" s_period="50"
                    o_period="600" aa_period="50" dev_type="DAC2_716" slope="6553.0”
                    intercept="0" min="0" max="10" step="0.0"/>
    </device>
</Logical_Pts>
```

MIB Framework Software

- Data Logging
  - Alerts and archive data
  - Sent via Multicast
  - transfer rate adjustable on a point by point basis

MIB Framework Software

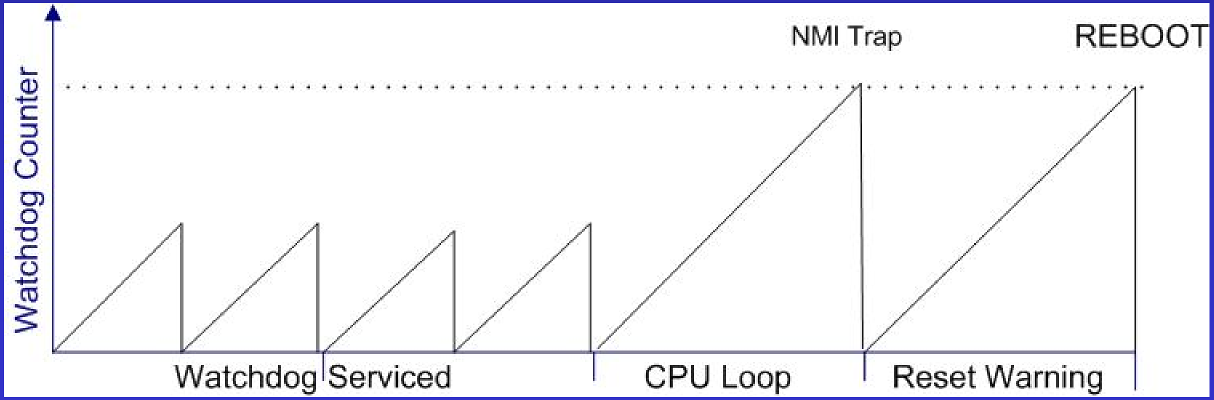


MIB Framework Software

- Reliability
  - – Recovery from SW Failure
  - – Watchdog timer
  - – Tracing cause of failure

MIB Framework Software

- Reliability (cont)



Module Specific Software

- Contained in module specific directory
- H/W details defined in table within ptsmon_usr_init.c
- LP definitions in logical_pt.xml
- Can be developed in < 1 day

Module Specific Software

- ptsmon_usr_init.c (example)

```
/*
 * Declare database used to gather data from module hardware
 */
TS_RAW_MON_PT Raw_Monitor_Points[] =
{
  {"rAGCV",    0,1,{AGCV_MON,0,0x0FFF,0},ADC_TLV2556,0,4,0,FIVE_SEC,0,NU_NULL},
  {"rPWR",     0,1,{PWR_MON,0,0x0FFF,0},ADC_TLV2556,0,4,1,FIVE_SEC,0,NU_NULL},
  {"rFMV1",    0,1,{FMV1_MON,0,0x0FFF,0},ADC_TLV2556,0,4,2,FIVE_SEC,0,NU_NULL},
  {"rYIGTemp",0,1,{YIGTEMP_MON,0,0x0FFF,0},TEMP_MAX66XX,0,6,1,FIVE_SEC,0,NU_NULL},
/* Note: the digital points come from Parallel I/O */
  {"rDIG_IO",  0,2,{{LOCK1_MON,0,0x3FFFFF,21},
                    {LOCK2_MON,0,0x7FFFFF,22}},GPIO,0,1,0,FIVE_SEC,0,NU_NULL},
  {NU_NULL}
};

int raw_pt_cnt = sizeof(Raw_Monitor_Points)/sizeof(Raw_Monitor_Points[0]);
```

Module Specific Software

- Any module specific procedural logic declared in module_specific_routines.c

Questions?