IPMItool Raw Command Interface to OpenPower POWER8 On Chip Controller: Sensor reading commands

Version: 0.4

Maintained by: Wael El-Essawy
welessa@us.ibm.com

Copyright and Disclaimer

IBM Systems and Technology Group 2070 Route 52, Bldg. 330 Hopewell Junction, NY 12533-6351

The IBM home page can be found at ibm.com®.

Overview

This document presents the IPMI interface to the OpenPower POWER8 on-chip controller (OCC). First, The raw command of the ipmitool is presented to issue OCC commands, then a detailed description is provided to issue Amester commands to get OCC sensor readings.

The OCC has to be enabled, and running in active state in order for the ipmitool commands to function properly. The first section in this document explains important procedures to check for the OCC state, and set it to active state, if it is not. Section 2 presents the ipmitool raw command format of general OCC commands. Section 3 presents the ipmitool raw command format for Amester call-through commands, and provides a set of examples that explain a common scenario to obtain OCC sensor readings.

1. Checking if the OCC is Enabled:

In order for ipmitool commands to provide requested functionality, the OCC has to be running in active state. If the OCC is not running in active state, the ipmitool will respond with the following error:

Unable to send RAW command (channel=0x0 netfn=0x3a lun=0x0 cmd=0xd rsp=0x90): Unknown (0x90)

The ipmitool provides a POWER8 sensor named "***OCC Active***" in single chip systems, which, when probed, provides the OCC State Information. For multi-chip POWER8 systems, there is an "***OCC N Active***" sensor for each OCC.

In order to probe a POWER8 systems for the state of each ot its OCC chips, you can use the following command:

ipmitool -H $bmc_ip -U $user -P $passwd ***sdr elist*** | *grep OCC*

Where:

- $bmc_ip: the BMC ip address
- $user: the BMC login name
- $passwd: the BMC password
- $occ_num: the occ number to which this command is directed (starts with 0x01)

On a single OCC system, the ipmitool response to this command will look like this:

```
OCC Active       | 08h | ok  | 210.0 | Device Enabled
```

Which indicates that this is a single chip system, and the OCC is running in active state.

In a multi-chip system (with multiple OCCs), the response may be like this:

```
OCC 1 Active     | 08h | ok  | 210.0 | Device Enabled
OCC 2 Active     | 09h | ok  | 210.1 | Device Enabled
```

Which indicates that this system has two OCCs, with active sensors “OCC 1 Active” and “OCC 2 Active”, i.e. N = 1 and N = 2 for the two OCCs OCC 1 and OCC 2.

On systems where the OCC is not enabled, the responses will look as follows:

```
single chip system:
OCC Active       | 08h | ok  | 210.0 | Device Disabled

multi-chip system:
OCC 1 Active     | 08h | ok  | 210.0 | Device Disabled
OCC 2 Active     | 09h | ok  | 210.1 | Device Disabled
```

2. IPMItool Raw command format interface to the Power8 OCC:

*ipmitool –H $bmc_ip –U $user –P $passwd raw **0x3a 0x0d $occ_num $cmd $payload***

Where:

- $bmc_ip: the BMC ip address
- $user: the BMC login name
- $passwd: the BMC password
- $occ_num: the occ number to which this command is directed (starts with 0x01)
- $cmd: the OCC command type
- $payload: an optional data field of the occ command

Note: Adding the “–v” switch as the first parameter to the ipmitool runs it in verbose mode, which prints useful information. Repeating the –v flag results in increasing the output debug level.

The impitool data return format:
CC B1 B2 B3 ..
Where:

- CC: The completion code of the ipmitool. E.g., “00” encodes success, any other value indicates otherwise, “0xC1” invalid command, etc. Consult the “ IPMI:

Intelligent Platform Management Interface Specification Second Generation v2.0" reference document for details.
- Bx: Optional data fields, whose format depends on the specific occ command

Please consult the "Interface Architecture TMGT to OCC" document for details on the $cmd and $payload fields, as well as the format of the returned data.

3. AME pass through command:

These commands are passed through the BMC to the Amester code on the OCC.

Format:
ipmitool –H $bmc_ip –U $user –P $passwd raw 0x3a 0x0d ***$occ_num $0x41 $ame_cmd 00 $ame_subcmd $ame_payload***

Where:
- $ame_cmd: the amester command. In POWER8 machines, the only two ame commands implemented are "0x3C" (amester api) and "0x3B" (amester manual throttle). If any other command is sent, the OCC amester code will respond with code "0xC1" (invalid command).
- $ame_subcmd: amester sub-command. Each Amester command has a list of subcommands, and optional $ame_payload data field.
- $ame_payload: optional parameters to the $ame_cmd and $ame_subcmd explained above.

The next table presents three sample Amester commands, followed by corresponding examples.

Amester command	Amester Sub-command	Payload
0x3C (Amester API)	0x07: Get Multiple Sensor Data	List of 16-bit sensor IDs
	0x21: Clear min/max fields in all sensors.	
	0x25: Get sensors info	$sensor_id $info_type

A Common scenario for collecting OCC sensors includes the following three steps:
1) Probe the OCC to get information about its sensors *($ame_subcmd* 0x025).
2) Clear the Min/Max fields of sensors.
3) Get a list of sensor readings.

The following three examples explain in details each of these three steps.
Examples:

Get sensors Info

Format:
ipmitool –H $bmc_ip –U $user –P $passwd raw 0x3a 0x0d 0x01 0x41 **0x3c** 0x00 **0x25** $sensor_id $info_type

To get information about a group of sensors, the Amester API command is used with the amester sub-command ($ame_cmd) 0x25. The Amester payload ($ame_payload) is comprised of two parts:

- $sensor_id: 2 bytes (16 bits) encoding the starting sensor ID. Amester responds with a list of the specified $info_type (explained in the next bullet) until the response buffer is full.
- $info_type: 1 byte encoding the field type to be returned by the OCC in the IPMI response packet. The following list provides all possible Sensor Type codes, along with an explanation of each type.

Note: Amester assumes the default buffer size to be 256 Bytes.

Field Type	Field Name	Description
0x00	NAME	Sensor Name “ASCII Code”
0x01	FREQ	Frequency of OCC sensor collection
0x02	UNITS	Unites of measurement
0x03	SCALE	Unit Scale of the sensor measurement. (Actual value = sensor reading ∗ unit scale in units of measurement).
0x05	ALL	All Data Types combined are returned

1. Get sensor names, starting with sensor 0x00:
ipmitool –H $bmc_ip –U $user –P $passwd raw 0x3a 0x0d 0x01 0x41 **0x3c** 0x00 **0x25** *0x00 0x00* 0x00

RAW RSP (243 bytes)
```
 00 41 4d 45 69 6e 74 64 75 72 00 41 4d 45 53 53
 64 75 72 30 00 41 4d 45 53 53 64 75 72 31 00 41
 4d 45 53 53 64 75 72 32 00 41 4d 45 53 53 64 75
 72 33 00 41 4d 45 53 53 64 75 72 34 00 41 4d 45
 53 53 64 75 72 35 00 41 4d 45 53 53 64 75 72 36
 00 41 4d 45 53 53 64 75 72 37 00 50 52 4f 42 45
 32 35 30 55 53 30 00 50 52 4f 42 45 32 35 30 55
 53 31 00 50 52 4f 42 45 32 35 30 55 53 32 00 50
 52 4f 42 45 32 35 30 55 53 33 00 50 52 4f 42 45
 32 35 30 55 53 34 00 50 52 4f 42 45 32 35 30 55
 53 35 00 50 52 4f 42 45 32 35 30 55 53 36 00 50
```

```
 52 4f 42 45 32 35 30 55 53 37 00 47 50 45 74 69
 63 6b 64 75 72 30 00 47 50 45 74 69 63 6b 64 75
 72 31 00 52 54 4c 74 69 63 6b 64 75 72 00 54 45
 4d 50 41 4d 42 49 45 4e 54 00 41 4c 54 49 54 55
 44 45 00
```

This is a list of 22 sensor names in zero terminated ASCII code. The following list includes the text format of this list:

{ “AMEintdur”, “AMESSdur0”, “AMESSdur1”, “AMESSdur2”, “AMESSdur3”, “AMESSdur4”, “AMESSdur5”, “AMESSdur6”, “AMESSdur7”, “PROBE250US0”, “PROBE250US1”, “PROBE250US2”, “PROBE250US3”, “PROBE250US4”, “PROBE250US5”, “PROBE250US6”, “PROBE250US7”, “GPEtickdur0”, “GPEtickdur1”, “RTLtickdur”, “TEMPAMBIENT”, “ALTITUDE”}

2. Get all Amester fields:

ipmitool –H $bmc_ip –U $user –P $passwd raw 0x3a 0x0d 0x01 0x41 **0x3c** 0x00 **0x25** *0x00 0x00* 0x05

```
> RAW RSP (238 bytes)
 00 41 4d 45 69 6e 74 64 75 72 00 75 73 00 00 00
 04 03 00 00 01 00 41 4d 45 53 53 64 75 72 30 00
 75 73 00 00 00 05 02 00 00 01 00 41 4d 45 53 53
 64 75 72 31 00 75 73 00 00 00 05 02 00 00 01 00
 41 4d 45 53 53 64 75 72 32 00 75 73 00 00 00 05
 02 00 00 01 00 41 4d 45 53 53 64 75 72 33 00 75
 73 00 00 00 05 02 00 00 01 00 41 4d 45 53 53 64
 75 72 34 00 75 73 00 00 00 05 02 00 00 01 00 41
 4d 45 53 53 64 75 72 35 00 75 73 00 00 00 05 02
 00 00 01 00 41 4d 45 53 53 64 75 72 36 00 75 73
 00 00 00 05 02 00 00 01 00 41 4d 45 53 53 64 75
 72 37 00 75 73 00 00 00 05 02 00 00 01 00 50 52
 4f 42 45 32 35 30 55 53 30 00 6e 2f 61 00 00 00
 04 03 00 00 01 00 50 52 4f 42 45 32 35 30 55 53
 31 00 6e 2f 61 00 00 00 04 03 00 00 01 00
```

This is the information related to 11 sensors, which are {sensor names, sensor collection frequency, units of measurements, and output scale}, as described in the table above. The following is a tabular form decoding the hexadecimal IPMI response buffer:

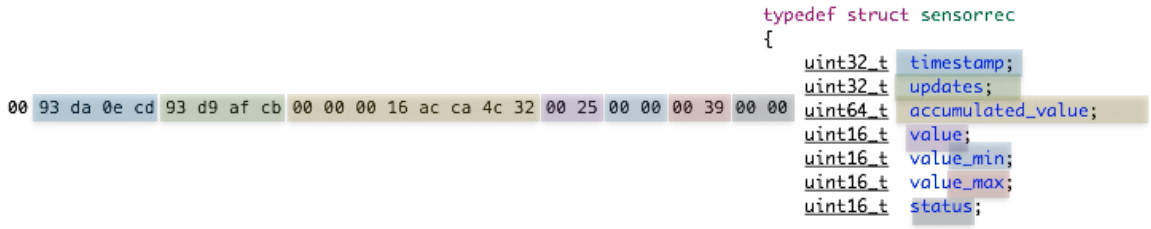
ID	Name	Unit	Frequency (kHz)	Scale
0x00	AMEintdur	us	4e3=4000	1
0x01	AMESSdur0	us	5e2=500	1
0x02	AMESSdur1	us	5e2=500	1
0x03	AMESSdur2	us	5e2=500	1
0x04	AMESSdur3	us	5e2=500	1
0x05	AMESSdur4	us	5e2=500	1
0x06	AMESSdur5	us	5e2=500	1

0x07	AMESSdur6	us	5e2=500	1
0x08	AMESSdur7	us	5e2=500	1
0x09	PROBE250US0	n/a	4e3=4000	1
0x0A	PROBE250US1	n/a	4e3=4000	1

3. read one sensor

ipmitool –H $bmc_ip –U $user –P $passwd raw 0x3a 0x0d 0x01 0x41
0x3c 0x00 **0x07** 0X00 0X00
00 93 da 0e cd 93 d9 af cb 00 00 00 16 ac ca 4c
32 00 25 00 00 00 39 00 00

The following table presents the same sensor (sensor 0x00, from examples 2.1 and 2.2, this is the AMEintdur sensor) in tabular format, color coding the different fields returned by the IPMI command.



- The 32 bits time stamp is 0x93da0ecd. This is the OCC time in ***ticks*** (250 microseconds). Getting two consecutive time readings, the timestamp difference can be used to infer the time window, and/or infer sensor reading rates over time.
- The updates field captures the count of the number of '250 usec ticks' that have passed between updates to this sensor (used for time-derived sensor): 0x93d9afcb.
- This is an accumulated reading of the sensor (continues increading until an overflow to 0 occurs.): 0x00000016acca4c32.
- The value field (0x0025), presents the latest sensor reading sample.
- The value_min field (0x0000) is the minimum sensor sample reading over a collection period since last min/max reset (from the sensors info table, this period is a 2 ms for sensor AMEintdur).
- The value_max field (0x0039) is the maximum sensor sample reading over a collection period since last min/max reset.
- The status field (0x0000) encodes the sensor status and control fields. A non zero value indicates that at the time of reading the sensor there is a pending sensors reset not executed yet.

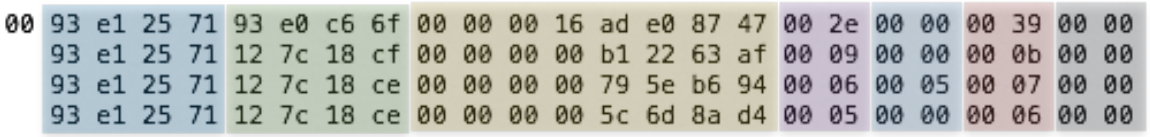
4. read a list of four sensors:

ipmitool –H $bmc_ip –U $user –P $passwd raw 0x3a 0x0d 0x01 0x41

0x3c 0x00 **0x07** 0X00 0X00 0X00 0X01 0X00 0X02 **0X00 0X03**

00 93 e1 25 71 93 e0 c6 6f 00 00 00 16 ad e0 87
47 00 2e 00 00 00 39 00 00 93 e1 25 71 12 7c 18
cf 00 00 00 00 b1 22 63 af 00 09 00 00 00 0b 00
00 93 e1 25 71 12 7c 18 ce 00 00 00 00 79 5e b6
94 00 06 00 05 00 07 00 00 93 e1 25 71 12 7c 18
ce 00 00 00 00 5c 6d 8a d4 00 05 00 00 00 06 00
00

and here is a layout of the ipmitool response data that decodes the different fields of the sensors:



The size of the data returned is 1+N*24, where N is the number of sensors read by this command

Note: The *accumulated_value* in older OCC firmware versions (prior to 840) is 32 bits only (4 bytes) rather than 64 bits, and the response packet should be formatted accordingly.

5. reset min max fields of all sensors

This command resets the min and max values of all OCC sensors.

ipmitool –H $bmc_ip –U $user –P $passwd raw 0x3a 0x0d 0x01 0x41
0x3c 0x00 **0x21**
00

The return packet is a single byte, which encodes the IPMI status

Appendix: POWER8 sensors Table

The Following table presents all POWER8 sensors, with all sensor information, global sensor ID (GSID) and a brief description.

It should be noted that not all sensors are available on all POWER8 systems. Based on your system configuration, and firmware level, only a subset of the sensors will be available.

The Sensors list for a specific OpenPower POWER8 system can be obtained by applying the get sensor command repeatedly until all sensors are collected. Please consult section 3, examples 1 and 2 for detailed instructions on getting sensors information presented in this table.

GSID	Sen. Name	Sen. #	Unit	Type	Loc	Samp. time	Scale Factor	Description
0x000	AMEintdur	1	us	TIME	OCC	250US	1	Time duration of AMEC IRQ (out of 250usec maximum window)
0x001-0x008	AMESSdurx	8	us	TIME	OCC	2MS	1	Time duration of internal AMEC state x (where x=0,1,..7)
0x009-0x010	PROBE250USx	8	n/a	GENERIC	OCC	250US	1	Internal probe sensor x (where x=0,1,..7)
0x011-0x012	GPEtickdurx	2	us	TIME	OCC	250US	1	Time duration spent in GPE Engine x (where x=0 and 1)
0x013	RTLtickdur	1	us	TIME	OCC	250US	1	Duration on the RTL tick interrupt
0x014	TEMPAMBIENT	1	C	TEMP	SYS	2MS	1	Ambient Temp of System (from APSS)
0x015	ALTITUDE	1	m	TEMP	SYS	2MS	1	Altitude of system
0x016	PWR250US	1	W	POWER	SYS	2MS	1	Bulk power of the system - Master only sensor
0x017	PWR250USFAN	1	W	POWER	SYS	2MS	1	Power consumption of the system fans - Master only sensor
0x018	PWR250USIO	1	W	POWER	SYS	2MS	1	Power consumption of the IO subsystem (including storage digital 5Volt or less rail) - Master only sensor
0x019	PWR250USSTORE	1	W	POWER	SYS	2MS	1	Power consumption of the storage subsystem (storage 12Volt rail) - Master only sensor
0x01A	PWR250USGPU	1	W	POWER	SYS	2MS	1	Power consumption of the GPU
0x01B	FANSPEEDAVG	1	RPM	TEMP	SYS	2MS	1	Average of all non-zero fan speeds - Master only sensor
0x01C-0x02B	PWRAPSSCH0	16	W	POWER	SYS	2MS	1	Power Provided by APSS channel x (where x=0, 1,..15)
0x2C	TODclock0	1	us	TIME	ALL	2MS	16	Time of day clock derived from a 32 MHz clock, Low order, 16 usec unit
0x2D	TODclock1	1	sec	TIME	ALL	2MS	0.1048576	TOD - Mid order, tenth of sec
0x2E	TODclock2	1	day	TIME	ALL	2MS	0.795364	TOD - high order, 0.8 days
0x2F	FREQA2MSP0	1	MHz	FREQ	PROC	2MS	1	Average of all core frequencies for

								Processor
0x30	IPS2MSP0	1	MIP	PERF	PROC	2MS	1	Vector sensor that takes the average of all the cores in Processor x
0x31	MEMSP2MSP0	1	%	TIME	PROC	2MS	1	Vector sensor that takes the minimum of all the channel-pairs in this Processor
0x32	PWR250USP0	1	W	POWER	PROC	250US	1	Power consumption for this Processor
0x33	PWR250USVDD0	1	W	POWER	PROC	250US	1	Power consumption for this Processor's Vdd Regulator Input (12Volt)
0x34	CUR250USVDD0	1	A	CURRENT	PROC	250US	0.01	Current consumption at Processor's Vdd Regulator Output
0x35	PWR250USVCS0	1	W	POWER	PROC	250US	1	Power consumption for this Processor's Vcs Regulator Input (12Volt)
0x36	PWR250USMEM0	1	W	POWER	PROC	250US	1	Power consumption for Memory for this Processor
0x37	SLEEPCNT2MSP0	1	#	PERF	PROC	2MS	1	Count the number of cores that are sleep in this processor
0x38	WINKCNT2MSP0	1	#	PERF	PROC	2MS	1	Count the number of cores that are winkled in this processor
0x39	SP250USP0	1	%	FREQ	PROC	250US	1	Percentage of Fmax (for current mode) that OCC is requesting
0x3A	TEMP2MSP0	1	C	TEMP	PROC	2MS	1	Vector sensor that is the average of all core temperatures for this Processor
0x3B	TEMP2MSP0PEAK	1	C	TEMP	PROC	2MS	1	Vector sensor that is the peak of all core temperatures for this Processor
0x3C	UTIL2MSP0	1	%	UTIL	PROC	2MS	0.01	Average of all core utilizations for this Processor (where 100% = fully utilized)
0x3D	VRFAN250USPROC	1	pin	TEMP	VRM	250US	1	VRFAN signal for all processors in the system
0x3E	VRHOT250USPROC	1	pin	TEMP	VRM	250US	1	VRHOT signal for all processors in the system (regulator overheating)
0x03f–0x04a	FREQ250USP0Cy	12	MHz	FREQ	CORE	250US	1	Requested frequency from OCC for Core y (where y = 1,2,..6, 9,10, ..14)
0x04b-0x056	FREQA2MSP0Cy	12	MHz	FREQ	CORE	2MS	1	Average/actual frequency for this processor, Core y based on OCA data (where y = 1,2,..6, 9,10, ..14)
0x057-0x062	IPS2MSP0Cy	12	MIP	PERF	CORE	2MS	1	Instructions per second for core y on this Processor (where y = 1,2,..6, 9,10, ..14)
0x063-0x06e	NOTBZE2MSP0Cy	12	cyc	PERF	CORE	2MS	1	Not Busy (stall) cycles counter for core y on this Processor (where y = 1,2,..6, 9,10, ..14)
0x06f-0x07a	NOTFIN2MSP0Cy	12	cyc	PERF	CORE	2MS	1	Not Finished (stall) cycles counter for core y on this Processor (where y = 1,2,..6, 9,10, ..14)
0x07b-0x086	SPURR2MSP0Cy	12	%	PERF	CORE	2MS	1	SPURR scale factor for this Processor, Core y: 100% corresponds to nominal frequency (where y = 1,2,..6, 9,10, ..14)

0x087–x092	TEMP2MSP0Cy	12	C	TEMP	CORE	2MS	1	Average temperature of DTS sensors for Processor's Core y: 100% corresponds to nominal frequency (where y = 1,2,..6, 9,10, ..14)
0x093-09e	UTIL2MSP0Cy	12	%	UTIL	CORE	2MS	0.01	Utilization of this Processor's Core y (where 100% = fully utilized): NOTE: per thread HW counters are combined as appropriate to give this core level utilization sensor (where y = 1,2,..6, 9,10, ..14)
0x09f-0x0aa	NUTIL3SP0Cy	12	%	UTIL	CORE	3S	0.01	Normalized average utilization of this Processor's Core y (where y = 1,2,..6, 9,10, ..14)
0x0ab-0x0b6	MSTL2MSP0Cy	12	cpi	PERF	CORE	2MS	1	Memory (L1) stalled cycles per instruction for this Processor, Core y (where y = 1,2,..6, 9,10, ..14)
0x0b7-0xc2	CMT2MSP0Cy	12	%	PERF	CORE	2MS	1	Core-level main memory access throttle setting for core y on this processor
0x0c3-0x0ce	CMBW2MSP0Cy	12	GBs	PERF	CORE	2MS	0.00256	Average Memory Band width for core y on this processor
0x0cf-0x0da	PPICP0Cy	12	%	PERF	CORE	2MS	1	Percentage of Prevented Instruction Cycles due to Throttling Events on core y
0x0db-0x0e6	PWRPX250USP0C	12	W	POWER	CORE	250US	1	Power Proxy sensor for Core y on this processor
0x0e7	UVOLT250USP0V0	1	mV	VOLTAGE	VRM	250US	0.1	Undervolting voltage request for this Processor's Vdd voltage rail
0x0e8	UVOLT250USP0V1	1	mV	VOLTAGE	VRM	250US	0.1	Undervolting voltage request for this Processor's Vcs voltage rail
0x0e9	VOLT250USP0V0	1	mV	VOLTAGE	VRM	250US	0.1	Voltage request for this Processor's Vdd voltage rail
0x0ea	VOLT250USP0V1	1	mV	VOLTAGE	VRM	250US	0.1	Voltage request for this Processor's Vcs voltage rail
0x0eb	VRFAN250USMEM	1	pin	TEMP	VRM	250US	1	VRFAN signal for all memory controllers in the system
0x0ec	VRHOT250USMEM	1	pin	TEMP	VRM	250US	1	VRHOT signal for all memory controllers in the system (overcurrent limit hit or regulator overheating)
0x0ed-0x0f4	MRD2MSP0Mx	8	GBs	PERF	MEM	2MS	0.00128	Memory read requests per sec for Processor's Memory Controller x
0x0f5-0x0fc	MWR2MSP0Mx	8	GBs	PERF	MEM	2MS	0.00128	Memory write requests per sec for Processor's Memory Controller x
0x0fd-0x104	TEMPDIMMAXP0Mx	8	C	TEMP	MEM	2MS	1	Hottest DIMM temperature behind Centaur x for this Processor
0x105-0x00c	LOCDIMMAXP0Mx	8	loc	TEMP	MEM	2MS	1	Location of the hottest DIMM temperaure behind Centaur x for this Processor
0x10d-0x11c	MAC2MSPxMy	16	rps	PERF	MEM	250US	1	Memory activation requests per sec for Processor's Memory Controller x, Centaur y, Port-pair z (MBA01/MBA23)
0x11d-0x12c	MPU2MSPxMy	16	rps	PERF	MEM	250US	1	Memory power-up requests per sec for Processor's Memory Controller x, Centaur y, Port-pair z

								(MBA01/MBA23)
0x12d-0x13c	MIRB2MSPxMy	16	eps	PERF	MEM	250US	1	Memory Inter-request arrival idle intervals, Processor's MC x, Centaur y, Port-pair z (MBA01/MBA23)
0x13e-0x14c	MIRL2MSPxMy	16	eps	PERF	MEM	250US	1	Memory Inter-request arrival idle intervals longer than low threshold, Processor 's MC x, Centaur y, Port-pair z (MBA01/MBA23)
0x14d-0x15c	MIRM2MSPxMy	16	eps	PERF	MEM	250US	1	Memory Inter-request arrival idle intervals longer than medium threshold, Processor's MC x, Centaur y, Port-pair z (MBA01/MBA23)
0x15d-0x16c	MIRH2MSPxMy	16	eps	PERF	MEM	250US	1	Memory Inter-request arrival idle intervals longer than high threshold, Processor's MC x, Centaur y, Port-pair z (MBA01/MBA23)
0x16d-0x17c	MTS2MSPxMy	16	cnt	PERF	MEM	250US	1	Last received Timestamp (frame count) from Centaur
0x17d-0x18c	MEMSP2MSPxMy	16	%	PERF	MEM	250US	1	Processor socket-level (summary) main memory throttle-level setting
0x18d-0x19c	M4RD2MSPxMy	16	GBs	PERF	MEM	2MS	0.00128	Memory cached (L4) read requests per sec for Processor's MC x, Centaur y, Port-pair z (MBA01/MBA23)
0x19d-1ac	M4WR2MSPxMy	16	GBs	PERF	MEM	2MS	0.00128	Memory cached (L4) write requests per sec for Processor's MC x, Centaur y, Port-pair z (MBA01/MBA23)
0x1ad-0x1b4	MIRC2MSP0Mx	8	eps	PERF	MEM	250US	1	Memory Inter-request arrival idle interval longer than programmed threshold at this Processor's MCs, Centaur x (centaur as a whole)
0x1b5-0x1bc	MLP2P0Mx	8	eps	PERF	MEM	250US	1	Number of LP2 exits at this Processor's MCs, Centaur x
0x1bd	TEMP2MSCENT	1	C	TEMP	MEM	250US	1	Hottest Centaur temperature for this processor
0x1be	TEMP2MSDIMM	1	C	TEMP	MEM	250US	1	Hottest DIMM temperature for this processor
0x1bf	MEMSP2MS	1	%	PERF	MEM	250US	1	Processor socket-level (summary) main memory throttle-level setting
0x1c0-0x1cb	UTIL2MSSLCG0z	12	%	UTIL	LPAR	2MS	0.00610352	Partition sensor y (where y = 00, 01, ..10, 11)