**Documentation for eps\_wrap.c**

**eps\_healthcheck():** Will return an array of uint32\_t values with a size of 4.

Returns:

|  |  |
| --- | --- |
| arrayOfFlags[0] | Returns i2c\_eps\_powerModuleStatus() |
| arrayOfFlags[1] | Returns i2c\_eps\_batteryModuleStatus() |
| arrayOfFlags[2] | Returns i2c\_eps\_getTelemetryGroup(0x01)  (aka solar panel temperature) |
| arrayOfFlags[3] | Returns i2c\_eps\_FDIRflag() |

All methods like i2c\_eps\_powerModuleStatus() will print statements of errors or flags for the user to read and will return the data that was sent from the I2C procedure. For example, if i2c\_eps\_powerModuleStatus() says there is only a 3V3 output error, it will print the statement “3V3 output error” then it will return the flag at the spot bit 0.

Note:

For the arrayOfFlags[2], the telemetry needs to be run through a calculation to convert the data into significant values. This goes the same for all of telemetry. The way to calculate and print the values is copied and pasted below.

**Telemetry Conversions:**

arrayOfFlags[2] holds every 16 bits to be a single calculation of data. So I used the BYTE16CAST to separate the different data and then used twos compliment and multiplied it by 0.5 as explained in the document.

***Most likely will only need this one:***

#define BYTE16CAST 0xFF

// twos comp done here

// data = the returned data from i2c procedure

double tm1 = (~(arrayOfFlags[2] & BYTE16CAST) + 1) \* 0.5;

double tm2 = (~((arrayOfFlags[2] >> 16) & BYTE16CAST) + 1) \* 0.5;

double tm3 = (~(arrayOfFlags[2] & BYTE16CAST) + 1) \* 0.5;

double tm4 = (~((arrayOfFlags[2] >> 16) & BYTE16CAST) + 1) \* 0.5;

double tm5 = (~(arrayOfFlags[2] & BYTE16CAST) + 1) \* 0.5;

PRINTF("M\_SP Temperature X+ = %d C \n", tm1);

PRINTF("M\_SP Temperature X- = %d C \n", tm2);

PRINTF("M\_SP Temperature Y+ = %d C \n", tm3);

PRINTF("M\_SP Temperature Y- = %d C \n", tm4);

PRINTF("M\_SP Temperature Z+ = %d C \n", tm5);

If there is a need for other telemetry data, check the eps\_wrap.c code and there should be the conversions calculated below.

**How to bitmask the data**

***Can copy and paste over the section. The print statements will get what each flag means.***

**arrayOfFlags[0]:** i2c\_eps\_powerModuleStatus()

if (arrayOfFlags[0] && (1 << 0))

{

PRINTF("3V3 output error\n");

}

if (arrayOfFlags[0] && (1 << 1))

{

PRINTF("5V output error\n");

}

if (arrayOfFlags[0] && (1 << 2))

{

PRINTF("12V output error\n");

}

if (arrayOfFlags[0] && (1 << 8))

{

PRINTF("PDM1 error\n");

}

if (arrayOfFlags[0] && (1 << 9))

{

PRINTF("PDM2 error\n");

}

if (arrayOfFlags[0] && (1 << 10))

{

PRINTF("PDM3 error\n");

}

if (arrayOfFlags[0] (1 << 11))

{

PRINTF("PDM4 error\n");

}

if (arrayOfFlags[0] && (1 << 12))

{

PRINTF("PDM5 error\n");

}

if (arrayOfFlags[0] && (1 << 13))

{

PRINTF("PDM6 error\n");

}

**arrayOfFlags[1]:** i2c\_eps\_batteryModuleStatus()

if (arrayOfFlags[1] && (1 << 0))

{

PRINTF("CC, Charge Control Flag. Set if battery charge is disabled.\n");

}

if (arrayOfFlags[1&& (1 << 1))

{

PRINTF("DC, Discharge Control Flag. Set if battery discharge is disabled.\n");

}

if (arrayOfFlags[1] && (1 << 2))

{

PRINTF(" CHGTF, Charge-Termination Flag. Set if battery is fully charged.\n");

}

if (arrayOfFlags[1] && (1 << 4))

{

PRINTF("SEF, Standby-Empty Flag. Set if capacity is below 10%%, unset if above 15 %%.\n");

}

if (arrayOfFlags[1] && (1 << 8))

{

PRINTF("Set if heater is active.\n");

}

if (arrayOfFlags[1] && (1 << 12))

{

PRINTF("Set if battery balancing is happening from top cell to bottom cell.\n");

}

if (arrayOfFlags[1] && (1 << 13))

{

PRINTF("Set if battery balancing is happening from bottom cell to top cell.\n");

}

**arrayOfFlags[2]:**

**Needs to use the telemetry conversions because it returns doubles (Go to telemetry conversions)**

**arrayOfFlags[3]:** i2c\_eps\_FDIRflag()

if (arrayOfFlags[3] && (1 << 0))

{

PRINTF("0 Set if last was command unknown\n");

}

if (arrayOfFlags[3] && (1 << 1))

{

PRINTF("Set if last command parameter was invalid.\n");

}

if (arrayOfFlags[3] && (1 << 2))

{

PRINTF("Set if watchdog was triggered.\n");

}

if (arrayOfFlags[3] && (1 << 3))

{

PRINTF("Set if BOR was triggered.\n");

}

if (arrayOfFlags[3] && (1 << 5))

{

PRINTF("Set if battery manager is unavailable.\n");

}

if (arrayOfFlags[3] && (1 << 6))

{

PRINTF("Set if VBAT1\_ADC is out of range.\n");

}

if (arrayOfFlags[3] && (1 << 7))

{

PRINTF("Set if VBAT2\_ADC is out of range.\n");

}

if (arrayOfFlags[3] && (1 << 8))

{

PRINTF("Set if IBAT\_BM is out of range.\n");

}

if (arrayOfFlags[3] && (1 << 9))

{

PRINTF("Set if TEMP\_BM is out of range.\n");

}

if (arrayOfFlags[3] && (1 << 10))

{

PRINTF("Set if TEMP\_MB is out of range.\n");

}

if (arrayOfFlags[3] && (1 << 11))

{

PRINTF("Set if TEMP\_DB1 is out of range.\n");

}