DSM Device

Script Library Functions

Configure DSM Parameters

-- Configure DSM core kernel operating parameters --

unsigned char **DSM_ConfigParam(**unsigned char* *pDescrip*, unsigned char *SensorFrameTime*,
unsigned char *HostPollRate*,
unsigned char *HostDwellTime***)**

Return Value: 0 = Success, 1 = General Fault

pDescrip: String pointer indicating DSM unit name : must be 16 char long
SensorFrameTime: Frame time interval between DSM sensor sampling : 1 s/quanta : 59sec max
HostPollRate: Number of sensor frames between host query polls : 1 frame/quanta
HostDwellTime: Time to dwell during host poll and listen for a host query : 5 ms/quanta : 800ms max

Standardized Sensor Configuration

-- Configure standard sensor parametrics for a DSM bridge or accelerometer channel --

unsigned char **DSM_ConfigSensor(**unsigned char *Chan*, unsigned char *Type*,
unsigned char *Model*,
unsigned char *Range*,
unsigned char *Specifics***)**

Return Value: 0 = Success, 1 = General Fault

Chan: DSM channel selection[1]
Type: Standard sensor type[1]
Model: Standard sensor model[1]
Range: Formulated sensor range[1] : "P" = + : "M" = - : "Z" = 0
: "C" = DegC : UE = microstrain : PSI = lbs/in^2 : MV = millivolt
: "G" = G-force : "MG" = milliG-force : "MS" = millisecond

Specifics: Specific sensor configuration[1]

SSPEC_ACCEL_PEAKG = DSM measures peak positive G-Force
SSPEC_ACCEL_FUNDFREQ = DSM measures vibration fundamental frequency

SSPEC_LASERTACH_ACORR = DSM uses autocorrelative technique to measure the average time interval between LaserTach pulses within the time window range

[1] See following configuration table

Chan	*Type*	*Model*	*Range*	*Specifics*
CHAN0	STYPE_DISABLED			
CHAN1 CHAN2 CHAN3 CHANX CHANY	STYPE_TC	SMODEL_TC_E SMODEL_TC_J SMODEL_TC_K SMODEL_TC_N SMODEL_TC_R SMODEL_TC_S SMODEL_TC_T	SRANGE_TC_M25P50C SRANGE_TC_M50P100C SRANGE_TC_M100P200C SRANGE_TC_M200P400C SRANGE_TC_M200P800C	
	STYPE_RTD	SMODEL_RTD_Pt100_385 SMODEL_RTD_Pt200_385 SMODEL_RTD_Pt100_3926 SMODEL_RTD_Ni120_672	SRANGE_RTD_M100P100C SRANGE_RTD_M200P400C	

Standardized Sensor Configuration (cont)

Chan	*Type*	*Model*	*Range*	*Specifics*
	STYPE_STRAIN	SMODEL_STRAIN_1KGF2	SRANGE_STRAIN_PM125UE SRANGE_STRAIN_PM250UE SRANGE_STRAIN_PM500UE	SSPEC_STRAIN_SING SSPEC_STRAIN_DUAL SSPEC_STRAIN_QUAD
	STYPE_PRESSURE	SMODEL_PRESSURE_OMEGAPX180B	SRANGE_PRESSURE_ZP40PSI SRANGE_PRESSURE_ZP100PSI	
	STYPE_PROX	SMODEL_PROX_OMEGAPRX102_18N		
	STYPE_ANALOG	SMODEL_ANALOG_BIPOLAR	SRANGE_ANALOG_PM100MV SRANGE_ANALOG_PM200MV SRANGE_ANALOG_PM500MV SRANGE_ANALOG_PM1000MV SRANGE_ANALOG_PM2500MV	
	STYPE_ACCEL	SMODEL_ACCEL_7251A_100 SMODEL_ACCEL_7251A_500 SMODEL_ACCEL_608A11	SRANGE_ACCEL_PM250MG SRANGE_ACCEL_PM500MG SRANGE_ACCEL_PM1G SRANGE_ACCEL_PM2G SRANGE_ACCEL_PM4G SRANGE_ACCEL_PM8G	SSPEC_ACCEL_PEAKG SSPEC_ACCEL_FUNDFREQ
	STYPE_LASERTACH	SMODEL_LASERTACH_TMS	SRANGE_LASERTACH_0MS SRANGE_LASERTACH_500MS SRANGE_LASERTACH_1000MS SRANGE_LASERTACH_1500MS	SSPEC_LASERTACH_ACORR

NOTE: Use NULLSEL definition for any non-applicable parameters

Customized Bridge Channel Sensor Configuration

-- Configure custom sensor parametrics for a DSM bridge channel --

unsigned char **DSM_ConfigCustomSensor(**unsigned char *Chan*, unsigned char *Model*,
unsigned char *BrConfig*,
unsigned char *BrStim*,
unsigned char *CC*,
unsigned char *Gain*,
unsigned char *PGAOffset*,
unsigned char *StabTime*,
unsigned char *SampAvg***)**

Return Value: 0 = Success, 1 = General Fault

Chan: DSM bridge channel selection[2]
Model: Custom model to receive all settings[2]
BrConfig: Configuration of channel bridge[2]
BrStim: Bridge stimulus[2]
CC: Constant current stimulus source setting[2]
Gain: Prog Gain Amp gain setting[2]
PGAOffset: Prog Gain Amp output offset voltage : 3/256 volts/quanta
StabTime: Sensor stabilization time delay : 1 ms/quanta
SampAvg: Number of samples to average

[2] See following configuration table

Chan	*Model*	*BrConfig*	*BrStim*	*CC*	*Gain*	
CHAN0	cCUSTOM0	cBYPASS	cDISABLE	cCC_0MA	cX0_5	cX50
CHAN1	cCUSTOM1	cQUARTER	cCC	cCC_1MA	cX1	cX100
CHAN2	cCUSTOM2	cHALF	cPM7_5	cCC_2MA	cX2	cX125
CHAN3	cCUSTOM3	cFULL	cP7_5GND	cCC_10MA	cX5	cX200
	cCUSTOM4	cPDOWN			cX10	cX250
	cCUSTOM5				cX20	cX500
	cCUSTOM6				cX35	cX1000
	cCUSTOM7					

NOTE: Use NULLSEL definition for any non-applicable parameters

Customized Accelerometer Channel Configuration

-- Configure custom parametrics for a DSM accelerometer channel --

```
unsigned char DSM_ConfigCustomAccel(unsigned char Chan, unsigned char Model,
                                                        unsigned char Specifics,
                                                        unsigned int SampFreq,
                                                        unsigned int NumSamp,
                                                        unsigned char Window,
                                                        unsigned char Gain,
                                                        unsigned char PGAOffset,
                                                        unsigned int Sensitivity,
                                                        unsigned int SwCapFc,
                                                        unsigned int StabTime,
Return Value: 0 = Success, 1 = General Fault            unsigned char BlockAvg)
```

Chan: DSM accelerometer channel selection[3]
Model: Custom model to receive all settings[3]
Specifics: Specific accelerometer formulated parameter selection[1]
SampFreq: Sampling frequency for all channels
NumSamp: Number of samples
Window: Window used to prevent FFT leakage[3]
Gain: Prog Gain Amp gain setting[3]
PGAOffset: Prog Gain Amp output offset voltage : 3/256 volts/quanta
Sensitivity: Sensitivity of the ICP accelerometer : 1 mV/G/quanta
SwCapFc: Switched capacitor lowpass filter cutoff frequency : 1 Hz/quanta
StabTime: Sensor stabilization time delay : 1 ms/quanta
BlockAvg: Number of sample blocks to average

[3] See following configuration table

Chan	*Model*	*Window*	*Gain*	
CHANX	cCUSTOMA	cWINDOW_NONE	cX0_5	cX50
CHANY	cCUSTOMB	cWINDOW_HAMMING	cX1	cX100
	cCUSTOMC	cWINDOW_HANNING	cX2	cX125
	cCUSTOMD	cWINDOW_BLACKMAN	cX5	cX200
	cCUSTOME	cWINDOW_BARTLETT	cX10	cX250
	cCUSTOMF	cWINDOW_KAISER_A2	cX20	cX500
	cCUSTOMG		cX35	cX1000
	cCUSTOMH			

NOTE: Use NULLSEL definition for any non-applicable parameters

Sample PreConfigured Sensor

-- Return formulated parameter for a pre-configured sensor channel --

```
unsigned char DSM_SampleSensor(unsigned char Chan, float* pValue,
                                                   unsigned char* pUnits,
                                                   DSM_CHStatusT* pCHStatus)
```

Return Value: 0 = Success, 1 = Fault as described in *CHStatus*[4]

Chan: DSM channel selection[4]
pValue: Formulated sensor value pointer
pUnits: Formulated sensor value units pointer[4]
pCHStatus: Channel fault status pointer[4]

[4] See following configuration table

Chan	*Units*	*CHStatus*	
CHAN0	SUNIT_DISABLED	typedef union DSM_CHStatusTag	
CHAN1	SUNIT_UNITLESS	{	
CHAN2	SUNIT_BOOLEAN	unsigned char *byte*;	
CHAN3	SUNIT_SECOND	struct	
CHANX	SUNIT_VOLTMEAN	{	
CHANY	SUNIT_DEGREEC	unsigned int *Fault* :1;	Sensor Fault
	SUNIT_PSI	unsigned int *OR* :1;	OverRange
	SUNIT_STRAIN	unsigned int *UR* :1;	UnderRange
	SUNIT_HERTZ	unsigned int *spare* :5;	(== 0)
	SUNIT_GFORCE	}b;	
	SUNIT_VOLTRMS	}DSM_CHStatusT;	
	SUNIT_RPM		

Capture Raw Data Block

-- Sample and store a DSM multi-channel raw data block to DSM SRAM --

unsigned char **DSM_CaptureRDB(**unsigned char* *pDescrip*, unsigned char* *pSigNames*,
unsigned int *ChanEnb*,
unsigned int *SampFreq*,
unsigned long *SampCnt*,
unsigned char *Format***)**

Return Value: 0 = Success, 1 = General Fault

pDescrip: Null terminated string pointer indicating the capture name : string is 16 char max
pSigNames: Array of 6 null terminated strings pointer indicating the channel signal names : each string is 12 char max
ChanEnb: Inclusive OR of up to 6 channel defines as listed in configuration table[5]
SampFreq: Channel sampling frequency - all channels sampled at same rate : 1 Hz/quanta
SampCnt: Number of samples per channel
Format: Defines DSM A/D sample output resolution bit width and format[5]

[5] See following configuration table

ChanEnb	*Format*	
CH0ENB	UINT12	Unsigned 12-Bit Integer (0000 dddd dddd dddd)
CH1ENB	INT12	Signed 12-Bit Integer (ssss sddd dddd dddd)
CH2ENB	UFRAC12	Unsigned 12-Bit Fractional (dddd dddd dddd 0000)
CH3ENB	FRAC12	Signed 12-Bit Fractional (sddd dddd dddd 0000)
CHXENB		
CHYENB		

Get Time

-- Returns current time in seconds since 01Jan2000 --

unsigned long **DSM_GetTime(**void**)**

Return Value: Number of seconds since midnight 01 January 2000 : 1 s/quanta

Sleep

-- DSM enters low power mode for specified number of seconds --

unsigned char **DSM_Sleep(**unsigned int *SleepSeconds*, unsigned int *HostPollSeconds***)**

Return Value: 0 = Complete, 1 = USB/wireless host communication preempt

SleepSeconds: Number of seconds to sleep : 1 s/quanta
HostPollSeconds: Number of seconds between wireless host query polls : 1 s/quanta

NOTES: If sleep time completes, return value is 0
If sleep is preempted by a USB or wireless host query, return value is 1

If *HostPollSeconds* = 0, wireless host polling does not occur

Time Delay

-- Delay execution for specified number of milliseconds --

void **DSM_TimeDelay(**unsigned long *Milliseconds***)**

Milliseconds: Number of milliseconds to delay firmware instruction execution : 1 ms/quanta

Get Week Time

-- Return the current Day of Week, Hour and Minute --

unsigned char **DSM_GetWeekTime(**WeekTimeT* *pWeekTime***)**

Return Value: 0 = Success, 1 = General Fault

pWeekTime: Time of week structure pointer[6]

[6] See following configuration table

WeekTime		*DoW*
typedef struct WeekTimeTag {		NULLSEL (== 0) SUNDAY (== 1)
unsigned char *Minute*;	0 to 59	MONDAY (== 2)
unsigned char *Hour*;	0 to 23	TUESDAY (== 3)
unsigned char *DoW*;	See *Day* column	WEDNESDAY (== 4)
}WeekTimeT;		THURSDAY (== 5) FRIDAY (== 6) SATURDAY (== 7)

Sleep Until Alarm

-- DSM enters low power mode until weekly based time alarm --

unsigned char **DSM_SleepUntilAlarm(**WeekTimeT* *pAlarmTime***)**

Return Value: 0 = Complete, 1 = USB host communication preempt

pAlarmTime: Time of week alarm settings[6]

NOTES: If sleep time completes, return value is 0
If sleep is preempted by a USB host query, return value is 1

If *DoW* == NULLSEL, the day of week is ignored

Power DSM Sensors

-- Apply/Remove Power to DSM Sensors --

unsigned char **DSM_PowerSensors(**unsigned int *ChanEnb***)**

ChanEnb: Inclusive OR of up to 6 channel defines as listed in configuration table[5]

Return Value: 0 = Success, 1 = General Fault

Power LED Color

-- Specify color of DSM Power LED --

void **DSM_PowerLED(**unsigned char *Color***)**

Color: Desired Power LED color[7]

[7] See following configuration table

Color	
OFF	Off
RED	Red
YEL	Yellow
GRN	Green
TOGRED	Toggle Red
TOGYEL	Toggle Yellow
TOGGRN	Toggle Green

Wireless LED Color

-- Specify color of DSM Wireless LED --

void **DSM_WirelessLED(**unsigned char *Color***)**

Color: Desired Wireless LED color[7]

Global Variables

-- Globally available parameters updated by DSM core processing kernel --

Formulated sensor parameters are available via:
float *DSM_SensorValBuff.CH0*
float *DSM_SensorValBuff.CH1*
float *DSM_SensorValBuff.CH2*
float *DSM_SensorValBuff.CH3*
float *DSM_SensorValBuff.CHX*
float *DSM_SensorValBuff.CHY*

-- Globally available definitions used by DSM core processing kernel --

DSM_DESCRIPSIZE	Size of string containing DSM unit name
DSM_NUMCHAN	Total number of channels available on DSM unit
RDB_SIGNAMESIZE	Size of channel signal name string used within DSM_CaptureRDB() function
RDB_DESCRIPSIZE	Size of raw data block description string used within DSM_CaptureRDB() function

Script Requirements

-- Required definitions within script.h --

ENABLEKERNELSCRIPT	Set to TRUE/FALSE to enable/disable kernel script hook to run during DSM core kernel
ENABLEPOLLSCRIPT	Set to TRUE/FALSE to enable/disable poll script hook to run during Device Utility Sotware App connection

-- Required functions within script.c --

void ScriptKernelInit(void)	Executes at DSM powerup to initialize settings for the kernel script
void ScriptPollInit(void)	Executes at DSM powerup to initialize settings for the poll script
void ScriptKernel(void)	Script which runs during DSM core kernel execution
void ScriptPoll(void)	Script which runs during DSM connection to the Device Utility Software App