DSP Services Function Descriptions (version 1.0.1.0)

The DSP Services provide an interface between the DSP and a DS6XXXHPI daughtercard that allows a DSP application to use the digital I/O, analog inputs, and communications resources on the daughtercard. The DSK6XXXHPI daughtercard supports the TMS320C6713 and TMS320C6416T DSKs. The TMS320C6416T DSK requires daughtercard software version 1.0.1.0 or higher. You can use winDSK6 version 4.0.1.0 or higher to verify the daughtercard software version.

DSP Services rely on a single fixed memory location (0x01A007F8, defined in dsp_services.h) to establish communications between the daughtercard and the DSP software. This is a memory location in the EDMA RAM area that is not normally used by the EDMA. Be sure that your program does not access this location.

The functions that make up the DSP Services interface are:

- int StartHpiServices();
- void SetDigitalIoDirection(unsigned short);
- unsigned short ReadDigitalIo();
- void WriteDigitalIo(unsigned short);
- void EnableAnalog(unsigned short);
- unsigned short ReadAnalog(unsigned short);
- unsigned int ReadSerial(char*, unsigned int);
- unsigned int ReadUsb(char*, unsigned int);
- unsigned int WriteSerial(char*, unsigned int);
- unsigned int WriteUsb(char*, unsigned int);
- int IsHpiDataNew();
- void SetBaudRate(unsigned short);

Individual functions are described in detail on the following pages.

If your program using the DSP Services is being loaded/run with Code Composer Studio (CCS), please observe the following sequence to ensure the DSP's HPI port is properly configured.

1. Download the program using CCS, but do not run.
2. Press and release the reset button on the DSK6713HPI daughtercard.
3. Run the program from CCS.

> Note that depending on the specific DSK version and CCS version that you are using, you may find that you need to reverse the order of steps 1 and 2.

If you are using the HPI daughtercard to load the program (i.e. using winDSK6 or C6X_Control to load the program), you do not have to worry about this issue.

StartHpiServices

int StartHpiServices();

Return Value

Returns non-zero on success. A zero return value indicates an error communicating with the daughtercard.

Parameters

None.

Remarks

Initializes the DSP Services data structures on the DSK, and then signals the HPI daughtercard to enter DSP services mode.

When the daughtercard enters DSP Services mode, it changes the LED flash pattern to a 2-pulse pattern. To leave DSP Services mode, the daughtercard must be reset or power cycled.

SetDigitalIoDirection

void SetDigitalIoDirection(unsigned short mask);

Return Value

Returns non-zero on success. A zero return value indicates an error communicating with the daughtercard.

Parameters

mask

Each bit of the mask sets the direction of the corresponding bit of digital I/O. Set the mask bits to 1 to configure the bit as input, and to 0 to configure the bit as output. Any pins configured as analog inputs will override this setting, and are not available as digital I/O.

See Table 1 for information on bit assignments.

Remarks

This function must be called before calling ***StartHpiServices***. Calling it afterwards has no effect.

ReadDigitalIo

unsigned short ReadDigitalIo();

Return Value

Returns the current state of the digital pins configured as inputs. See Table 1 for information on bit assignments.

Parameters

None.

Remarks

This function reads the data structures on the DSK. The daughtercard updates this data from the digital I/O pins at a 1.0 kHZ rate. To know when an update has occurred, see ***IsHpiDataNew***.

WriteDigitalIo

void WriteDigitalIo(unsigned short mask);

Return Value

None.

Parameters

mask

Each bit of the mask sets the value of the corresponding bit of digital I/O that is configured as output. The bit value has no effect on pins configured as inputs.

See Table 1 for information on bit assignments.

Remarks

This function updates the data structures on the DSK. The daughtercard updates the digital I/O pins at a 1.0 kHZ rate using these values. To know when the update has occurred, see ***IsHpiDataNew***.

EnableAnalog

void EnableAnalog(unsigned short mask);

Return Value

None.

Parameters

mask

Enables the selected I/O pins for analog input. The least significant 8 bits of this parameter correspond to the D7:0 inputs on the parallel port connector. Set the bit to 1 to enable the pin for analog input. Any pins configured as analog inputs are not available as digital I/O.

See Table 1 for information on pin assignments for the analog inputs.

Remarks

This function must be called before calling ***StartHpiServices***. Calling it afterwards has no effect.

ReadAnalog

unsigned short ReadAnalog(unsigned short channel);

Return Value

Returns the most recent value read on the selected analog channel. If the selected analog channel has not been enabled, the return value is undefined.

Parameters

channel

Must be 0 though 7 to select the corresponding analog channel data. See Table 1 for information on pin assignments for the analog inputs.

Remarks

This function reads the data structures on the DSK. The daughtercard updates this data from the analog inputs at a 1.0 kHZ rate. To know when an update has occurred, see ***IsHpiDataNew***.

ReadSerial
ReadUSB

unsigned int ReadSerial(char *buffer, unsigned int size);
unsigned int ReadUsb(char *buffer, unsigned int size);

Return Value

Returns the number characters actually placed in the buffer.

Parameters

buffer

Pointer to a buffer that incoming data is to be written to. It is the caller's responsibility to ensure that the buffer is at least as large as the *size* parameter.

size

Indicates the size of the buffer in bytes. The function will not attempt to store more data than this value.

Remarks

These functions allow access to the serial and USB communications resources on the HPI daughtercard. This function actually reads the communication queues on the DSK, which are updated by the daughtercard at a 1.0 kHZ rate. To know when an update has occurred, see ***IsHpiDataNew***.

WriteSerial
WriteUSB

unsigned int WriteSerial(char *buffer, unsigned int size);
unsigned int WriteUsb(char *buffer, unsigned int size);

Return Value

Returns the number characters actually sent.

Parameters

buffer

Pointer to a buffer containing the data to send. It is the caller's responsibility to ensure that the buffer is at least as large as the *size* parameter.

size

Indicates the number of bytes to send from the buffer.

Remarks

These functions allow access to the serial and USB communications resources on the HPI daughtercard. This function actually writes to the communication queues on the DSK, which are then read by the daughtercard at a 1.0 kHZ rate. To know when an update has occurred, see ***IsHpiDataNew***.

IsHpiDataNew

int IsHpiDataNew();

Return Value

Returns non-zero if that daughtercard has updated the data structures on the DSK since the last time *IsHpiDataNew* was called. Returns zero otherwise.

Parameters

None.

Remarks

The daughtercard updates the DSK data structures at a 1.0 kHZ rate. A non-zero return value from this function indicates that an update has occurred since the last call to the function.

SetBaudRate

void SetBaudRate(unsigned short baud_divider);

Return Value

None.

Parameters

baud_divider

Baud rate divisor used to set the baud rate of the RS-232 port on the daughtercard. A number of values for common baud rates have been predefined in *hpi_services.h*:

HPI_SVC_BAUD115200
HPI_SVC_BAUD57600
HPI_SVC_BAUD38400
HPI_SVC_BAUD19200
HPI_SVC_BAUD9600
HPI_SVC_BAUD4800
HPI_SVC_BAUD2400
HPI_SVC_BAUD1200

Remarks

The actual baud rate is equal to 7.3728 MHz divided by *baud_divider*. Values other than those defined above can be used as long as they are within the allowable range for the MSP430 microcontroller on the daughtercard.

This function must be called before calling ***StartHpiServices***. Calling it afterwards has no effect.