Application Report
SPNA213–April 2015

Hercules™ SCI With DMA

Dave Livingston

ABSTRACT

This application report summarizes the necessary steps to setup the direct memory access controller (DMA) to transfer data between the SCI and the data RAM of the microcontroller, freeing the CPU during the entire message transmission. Detailed code examples are provided as guidelines for the different setup steps.

Project collateral and source code discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/spna213.

Contents

List of Figures

List of Tables

1 Introduction

The DMA controller is used to transfer data between two locations in the memory map in the background of CPU operations. This application report demonstrates how to use the DMA to free CPU cycles while transmitting SCI messages. The example was designed to be run on the LAUNCHXL2 series of launchpads and will transmit an incremented number totaling 1890 characters using SCI2 in multi-buffered mode. SCI2 is included in the debug interface USB connection and can be monitored with a standard terminal.

2 Basic Steps to Setup a DMA SCI Transfer

In order to use the DMA for data transfer between the SCI and the device data RAM, the module needs to be setup properly. The following bullets show the basic steps that need to be performed to do this. Each item is discussed in detail in the following sections of this document. In the example, this is completed by the *scidmaInit* function.

- Initialize the desired SCI
- Enable DMA
- Enable SCI DMA transfer interrupts (optional interrupt mode)
- Initialize DMA transfer group
- Set data transfer base address
- Setup transfer direction in the configuration RAM (TCR)
- Trigger DMA transfers
- Get transfer ready notification

2.1 DMA Configuration With scidmaInit()

Configure the DMA for the most efficient transfers allowed by the system and your application. Here is an example of the DMA configuration with SCI2 in multi-buffer mode for a transmission. Before scidmaInit is called, scInit and linsci2enableMBUFF should be called to configure SCI2 for multi-buffer mode. The DMA reads 32 bits at a time from the data RAM incrementing the address, writing 32 bits of data to the transmit buffer without incrementing the address. The SCI transmits one byte at a time until all 4 bytes have been transmitted, then signals that it is ready for more data either with a ready or request signal to the desired controller - in this case the DMA. Then, the DMA transfers the next 4 bytes until the entire message has been sent and sets a complete flag in the data RAM for the CPU to know when it is ready to send another message.

```
                                        /* Populate dma control packets structure */
g_dmaCTRLPKT.CHCTRL = 0;                /* channel control */
g_dmaCTRLPKT.ELCNT = 1;                 /* element count */
g_dmaCTRLPKT.ELDOFFSET = 0;             /* element destination offset */
g_dmaCTRLPKT.ELSOFFSET = 0;             /* element source offset */
g_dmaCTRLPKT.FRDOFFSET = 0;             /* frame destination offset */
g_dmaCTRLPKT.FRSOFFSET = 0;             /* frame source offset */
g_dmaCTRLPKT.PORTASGN = 4;              /* port b */
g_dmaCTRLPKT.RDSIZE = ACCESS_32_BIT;    /* read size */
g_dmaCTRLPKT.WRSIZE = ACCESS_32_BIT;    /* write size */
g_dmaCTRLPKT.TTYPE = FRAME_TRANSFER ;   /* transfer type */
g_dmaCTRLPKT.ADDMODERD = ADDR_INC1;     /* address mode read */
g_dmaCTRLPKT.ADDMODEWR = ADDR_FIXED;    /* address mode write */
g_dmaCTRLPKT.AUTOINIT = AUTOINIT_OFF;   /* autoinit */
```

Note that the peripheral bus does not allow for 64-bit unaligned accesses. When in multi-buffer mode, the largest accesses possible to the SCI TDx registers is 32 bits. Care should be taken to account for CPU endianness as the SCI has the register formatted in little endianism.

2.2 DMA SCI Transmit With scidmaSend()

Every time scidmaSend is called, it completes the control packet with the addresses and number of bytes to transfer.

```
                                                /* Populate dma control packets structure */
g_dmaCTRLPKT.SADD = (uint32)source_address;     /* source address */
g_dmaCTRLPKT.DADD = (uint32)(&(linREG->TDx));   /* destination address */
g_dmaCTRLPKT.FRCNT = strlen(source_address)/4+8; /* frame count */
```

Then scidmaSend calls the dmaSetCtrlPacket to update the dma module. Once this is complete, the scidmaSend configures the DMA to start the transfer freeing the CPU to do other work until the entire message has been transmitted. A flag has been added to the scidmaSend to prevent it from starting a new message until it is finished with the last message.

```
/** @fn void scidmaSend(char *source_address)
* @brief Initialize the SCI and DMA to tranfer SCI data via DMA
* @note This function configures the SCI to trigger a DMA request when the SCI TX is complete.
*
* This function configures the DMA for SCI2 in single buffer or multi-buffer mode.
* In single buffer mode
*                                                                    the DMA moves 1 Byte
to the SCI2 tranmit register when the request is set.
* In multi-buffer mode
*                                                                the DMA moves 4 Bytes
to the SCI2 transmit buffer when the request is set.
*/
void scidmaSend(char *source_address)
{
#if ((__little_endian__ == 1) || (__LITTLE_ENDIAN__ == 1))
 uint8 dest_addr_offset=0;      /* 0 for LE */
#else
 uint8 dest_addr_offset=3;      /* 3 for BE */
#endif

 /* Wait for the DMA to complete any existing transfers */
 while(DMA_Comp_Flag != 0x55AAD09E);

 /* Reset the Flag to not Done*/
 DMA_Comp_Flag = ~0x55AAD09E;

 /* - Populate dma control packets structure */
   g_dmaCTRLPKT.SADD       = (uint32)source_address;             /* source address            */

 if (((scilinREG->GCR1 >> 10U) & 1U) == 0U) {                   /* SCI2 multi-
buffer mode      */
   g_dmaCTRLPKT.DADD   = (uint32)(&(scilinREG->TD))+dest_addr_offset;
   g_dmaCTRLPKT.RDSIZE = ACCESS_8_BIT;                          /* read size                 */
   g_dmaCTRLPKT.WRSIZE = ACCESS_8_BIT;                          /* write size                */
   g_dmaCTRLPKT.FRCNT  = strlen(source_address);                /* frame count               */

 } else {
   g_dmaCTRLPKT.DADD      = (uint32)(&(linREG->TDx));
   g_dmaCTRLPKT.RDSIZE    = ACCESS_32_BIT;                      /* read size                 */
   g_dmaCTRLPKT.WRSIZE    = ACCESS_32_BIT;                      /* write size                */
   g_dmaCTRLPKT.FRCNT     = strlen(source_address)/4+8*         /* frame count               */
 }

 /* - setting dma control packets for transmit */
 dmaSetCtrlPacket(DMA_CH0,g_dmaCTRLPKT);

 /* - setting the dma channel to trigger on h/w request */
 dmaSetChEnable(DMA_CH0, DMA_HW);

 /* Enable TX DMA */
 scilinREG->SETINT = (1 << 16);

} /* scidmaSend */
```

3 HALCoGen Example

3.1 Introduction

This example demonstrates how to use the DMA to transfer a message to the SCI in multi-buffer mode for transmission. Without the DMA the CPU would have to move data to the SCI after every buffer sized bytes. With the DMA the CPU is free for the entire message transmission.

Project collateral and source code discussed in this application report were developed using HALCoGen 04.02.00 for the LaunchXL2 launchpad. This code was written to be easily ported to any HALCoGen project by copying sys_main.c, notification.c and sci.c. When generating a HALCoGen project, select any Hercules device, enable the drivers for the desired SCI, enable the necessary VIM channels such as 40 (DMA BTCA) and configure the desired SCI settings.

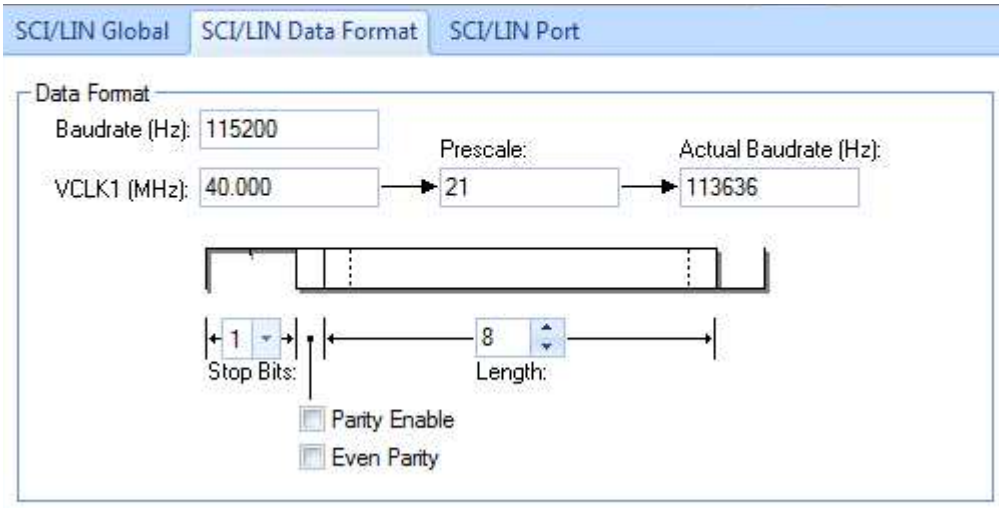

Figure 1. HALCoGen LaunchXL2 SCI2 Configuration 115200bps 81NN

On the LaunchXL2, SCI2 is the only SCI that supports multi-buffer transfers and is routed to the USB debug connector as XDS110 Class Application/User UART (COM#). Configure the terminal for a baud rate of 115.2 Kbps, 8 data bits, 1 stop bit, no parity, no flow control. Below is the expected output.

```
 Hercule SCI DMA Example - Version 00.00.01.07
******************************************************************************

scidmaSend Example - DMA to transfer single byte from RAM to the SCI
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123
124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195
196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219
220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243
244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267
268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291
292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315
316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339
340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363
364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387
388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411
412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459
460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483
484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499

scidmaSend Example Complete, 1890 characters sent
     CPU idle count: 1480577


scidmaSend Example - DMA to transfer four bytes from RAM to  SCI
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123
124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147
148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171
172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195
196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219
220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243
244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267
268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291
292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315
316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339
340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363
364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387
388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411
412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435
436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459
460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483
484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499

scidmaSend Example Complete, 1890 characters sent
     CPU idle count: 3359718


 sci transmit time @ 115200bps 8 data bits, 1 stop bit, no parity, no flow control (81NN)
                        ((1890 Bytes) * 10bits per byte ) / 115200:     0.164 S

 Demo Complete
******************************************************************************
```

NOTE: CPU idle count is a simple incremented variable in a polling loop and not CPU cycles. This could vary based on compile time options, if CPU cycles are desired, the PMU can be used to take an accurate measurement.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of significant portions of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed a special agreement specifically governing such use.

Only those TI components which TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components which have ***not*** been so designated is solely at the Buyer's risk, and that Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265