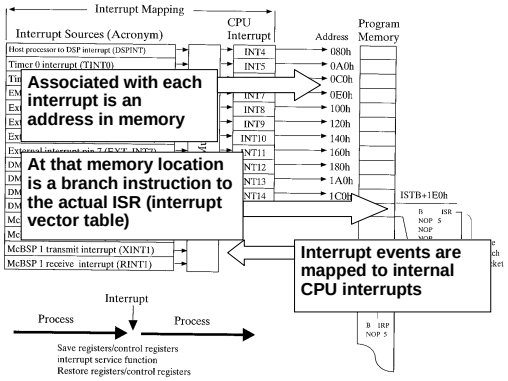
Lecture 4

- What are interrupts?
- What is CSL and BSL?
- Using AD535 codec
  - Output sine to DSK codec via serial port
- Using Interrupts
  - Use serial port interrupt to synchronize sinewave sample output

Interrupts

- An event that causes processor to halt what it is doing, and execute an interrupt service routine (ISR)
- Sources of interrupts include:
  - Timers
  - External interrupts
  - DMA (direct memory access)
  - McBSP transmit or receive

Interrupts



Setting up interrupts

- Assign events to CPU interrupts
- Create the interrupt vector table

CCS Configuration Tool

Interrupt service routines

1. Save registers (context save)
2. Actual function to perform
3. Restore registers (context restore)

Use Dispatcher in Configuration Tool

Enabling interrupts

- Individual interrupts can be turned on or off (set appropriate bits in Interrupt Enable Register)
- A master switch can be used to turn all interrupts on or off (Global Enable Interrupt bit in the Control Status Register)

These are defined in irq.h in the Chip Support Library (CSL)

TI Software Foundation Libraries

Board Support Library (BSL)
Higher-level routines supporting DSK-specific functionality
BSL routines make use of CSL routines

```
bsl_ad535.h
bsl_led.h
bsl_dip.h
...
```

Chip Support Library (CSL)
Low-level routines supporting on-chip peripherals

```
mcbsp.h
edma.h
irq.h
...
```

C Compiler Runtime Support Library
Standard ANSI C libraries

```
stdio.h
math.h
...
```

TI DSP

Peripheral Support Libraries

'C6000 CSL Modules	
CSL	CSL initialization
CHIP	Specify device
CACHE	Cache
DAT	Device-independent data copy/fill
DMA	Direct memory access
EDMA	Enhanced direct memory access
EMIF	External memory interface
HPI	Host port interface
IRQ	Interrupt controller
MCBSP	Multichannel buffered serial port
PWR	Power down
STDINC	Standard include
TIMER	Timer

'C6711 DSK BSL	
BSL	BSL initialization
BOARD	specify board
AD535	Access audio codec
DIP	Read board DIP switches
FLASH	Program FLASH ROM
LED	Write LED's

DSP/BIOS	
HWI	Hardware interrupts

General Procedure to use BSL/CSL

1. **Declare variables**
   - Usually handle & configuration

Example:

```
AD535_Handle myHandle;
AD535_Config myConfig = {
        AD535_LOOPBACK_DISABLE,
        AD535_MICGAIN_OFF,
        AD535_GAIN_0DB,
        AD535_GAIN_0DB };
```

General Procedure to use BSL/CSL

1. **Declare variables**
   - Usually handle & configuration
2. **Open peripheral**
   - Reserves resource
   - Provides 'handle' to reference resource (AD535_localId is a pre-defined pointer, specifically for the codec on the DSK)

Example:

```
AD535_Handle myHandle;
AD535_Config myConfig = {
        AD535_LOOPBACK_DISABLE, ... };

main() { myHandle = AD535_open(AD535_localId);
```

General Procedure to use BSL/CSL

1. **Declare variables**
   - Usually handle & configuration
2. **Open peripheral**
   - Reserves resource
   - Provides 'handle' to reference resource
3. **Configure peripheral**
   - Apply setup configuration to peripheral

Example:

```
AD535_Handle myHandle;
AD535_Config myConfig = {
        AD535_LOOPBACK_DISABLE, ... };

main() { myHandle = AD535_open(AD535_localId);
         AD535_config(myHandle, &myConfig);
```

General Procedure to use BSL/CSL

1. **Declare variables**
   - Usually handle & configuration
2. **Open peripheral**
   - Reserves resource
   - Provides 'handle' to reference resource
3. **Configure peripheral**
   - Apply setup configuration to peripheral
4. **Use peripheral**
   - Some periph's use functions to read/write them

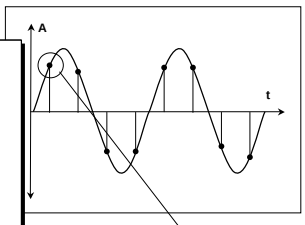
Example:

```
AD535_Handle myHandle;
AD535_Config myConfig = {
        AD535_LOOPBACK_DISABLE, ... };

main() { myHandle = AD535_open(AD535_localId);
         AD535_config(myHandle, &myConfig);
         AD535_write(myHandle, value);
```

Creating a Sine Wave

- Algorithm
- Real-time events
- Initialization

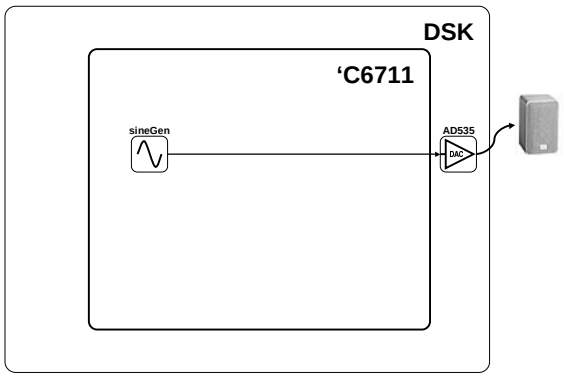
```
#define SAMPLE_FREQ 8000.0
#define SINE_FREQ 3000.0

int sinegen(void)
{
  float wave;
  int D=256;   /* size of wave table */
  int k;
  static int t=0;

  /**
  Code from Lab 3 to generate
  sine wave is here
  **/

  return((int)(32000*wave));
}
```



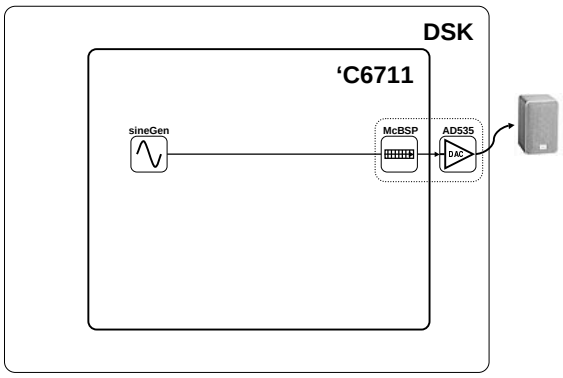
Sine_gen.c

Generates a value for each output sample

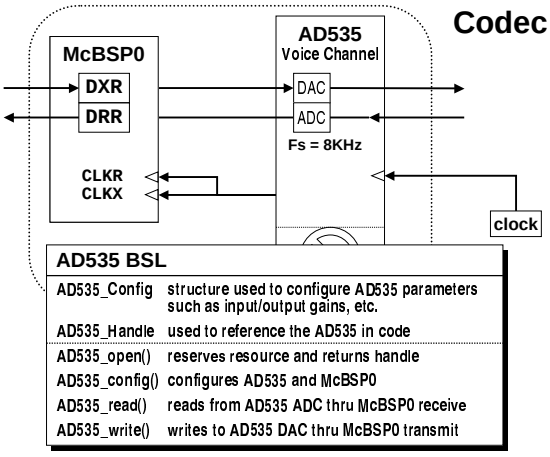
Output Sine via AD535 Codec



Send to Codec via McBSP

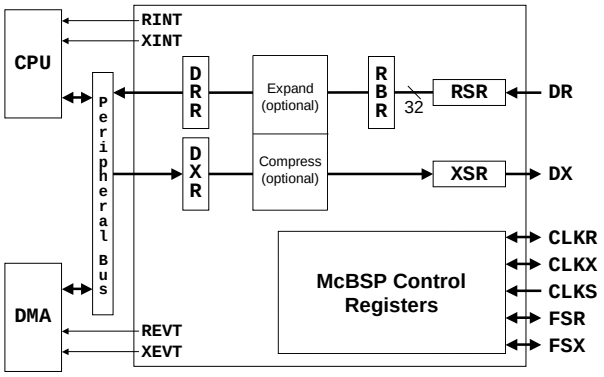


Codec

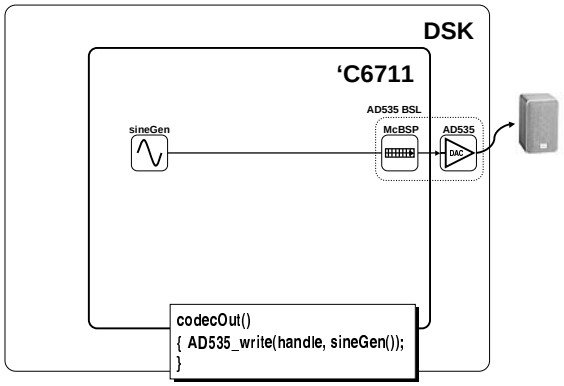


AD535 BSL	
AD535_Config	structure used to configure AD535 parameters such as input/output gains, etc.
AD535_Handle	used to reference the AD535 in code
AD535_open()	reserves resource and returns handle
AD535_config()	configures AD535 and McBSP0
AD535_read()	reads from AD535 ADC thru McBSP0 receive
AD535_write()	writes to AD535 DAC thru McBSP0 transmit

McBSP Block Diagram



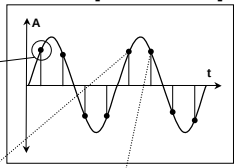
AD535_write



Synchronizing the Output Samples

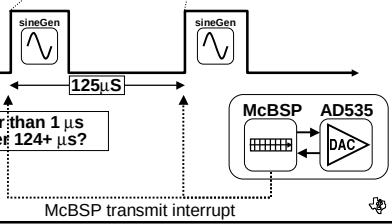
Algorithm

sineGen Generates a single sine sample



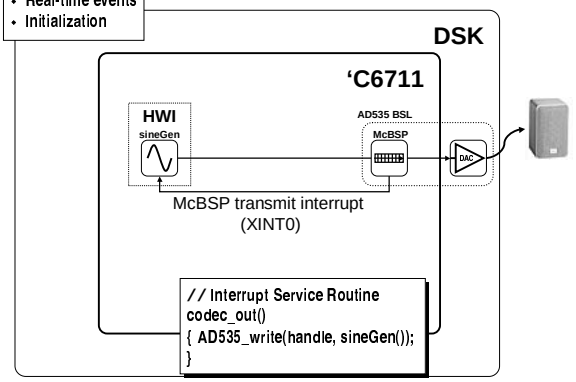
Timing

- How often should we output a sample?
- Since sineGen is faster than 1 µs how do we wait another 124+ µs?

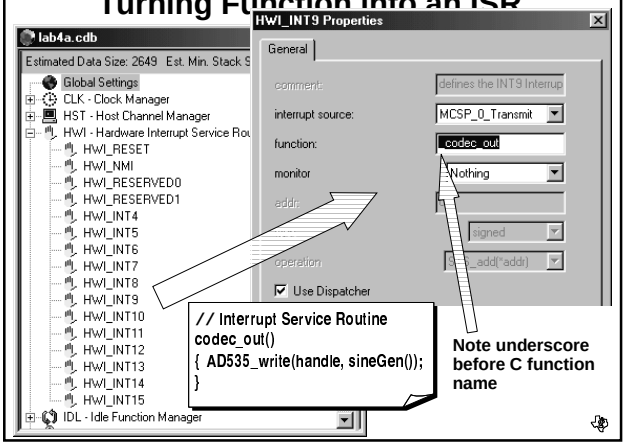


Real-time Sync to XINT0

- Algorithm
- Real-time events
- Initialization

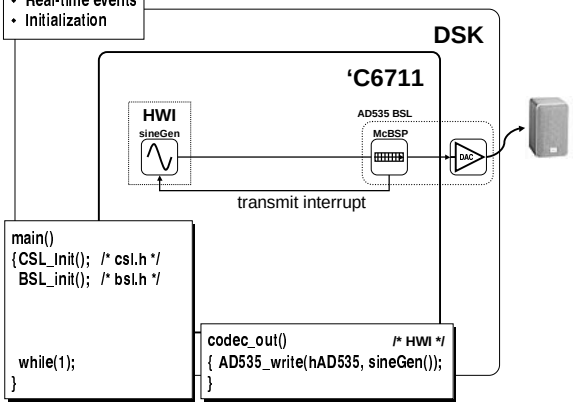


Turning Function into an ISR



Codec Initialization

- Algorithm
- Real-time events
- Initialization



Codec Initialization

- Algorithm
- Real-time events
- Initialization

```
/* BSL bsl_ad535.h */
AD535_Handle hAD535;
AD535_Config myAD535 = {
        AD535_LOOPBACK_DISABLE,
        AD535_MICGAIN_OFF,
        AD535_GAIN_0DB,
        AD535_GAIN_0DB };

void codec_init() {
        /* BSL bsl_ad535.h */
        hAD535 = AD535_open(AD535_localId);
        AD535_reset(hAD535);
        AD535_config(hAD535, &myAD535); }
```

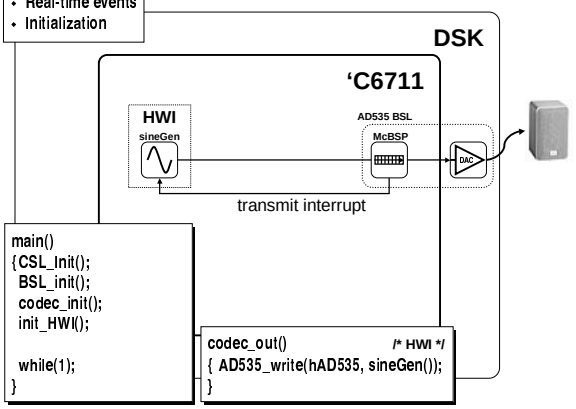
```
main()
{CSL_Init();
 BSL_init();
 codec_init();
 while(1);
}
```

Defined in bsl6711dsk.lib
(included in Linker:Options)

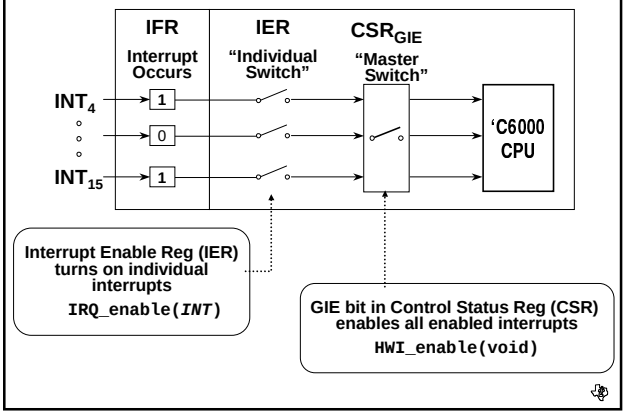
```
codec_out()                    /* HWI */
{ AD535_write(hAD535, sineGen());
}
```

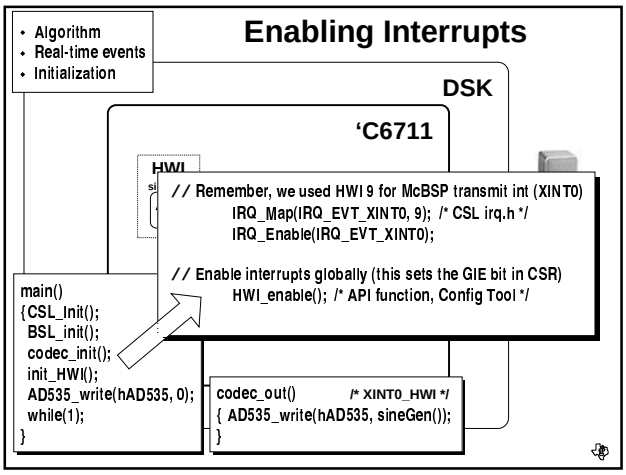
Enabling Interrupts

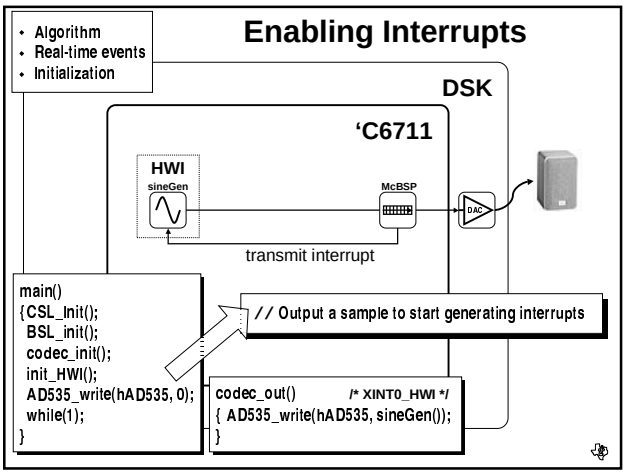
- Algorithm
- Real-time events
- Initialization



Enabling Interrupts




Enabling Interrupts
Algorithm
Real-time events
Initialization
DSK
'C6711
HWI
// Remember, we used HWI 9 for McBSP transmit int (XINT0)
IRQ_Map(IRQ_EVT_XINT0, 9); /* CSL irq.h */
IRQ_Enable(IRQ_EVT_XINT0);
// Enable interrupts globally (this sets the GIE bit in CSR)
HWI_enable(); /* API function, Config Tool */
main()
{CSL_Init();
BSL_init();
codec_init();
init_HWI();
AD535_write(hAD535, 0);
while(1);
}
codec_out() /* XINT0_HWI */
{ AD535_write(hAD535, sineGen());
}


Enabling Interrupts
Algorithm
Real-time events
Initialization
DSK
'C6711
HWI
sineGen
McBSP
DAC
transmit interrupt
main()
{CSL_Init();
BSL_init();
codec_init();
init_HWI();
AD535_write(hAD535, 0);
while(1);
}
// Output a sample to start generating interrupts
codec_out() /* XINT0_HWI */
{ AD535_write(hAD535, sineGen());
}