Embedded System Design Techniques™

Bootloader Design for MCUs

Session 3: Setting up a Test Application

January 27th, 2016

Jacob Beningo, CSDP

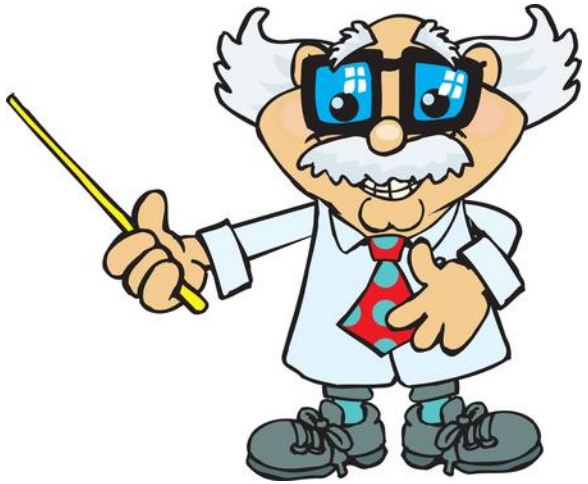
DesignNews

CEC CONTINUING EDUCATION CENTER

Presented by: Digi-Key ELECTRONICS

Session Overview

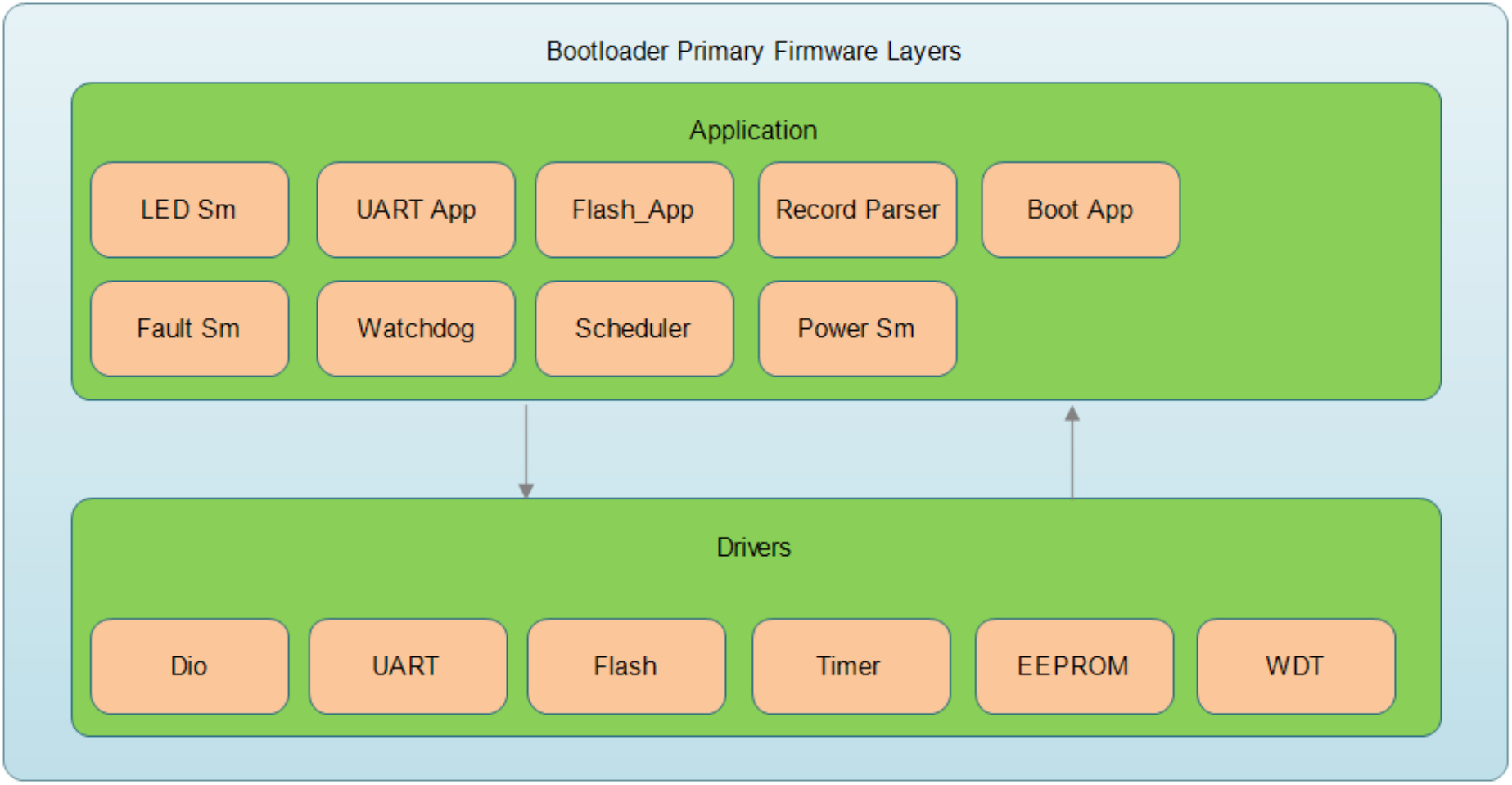
- Components
- Project Organization
- Api's and HAL's
- Accessing Flash
- Linker Files
- Command Parsing
- Watchdog Timers



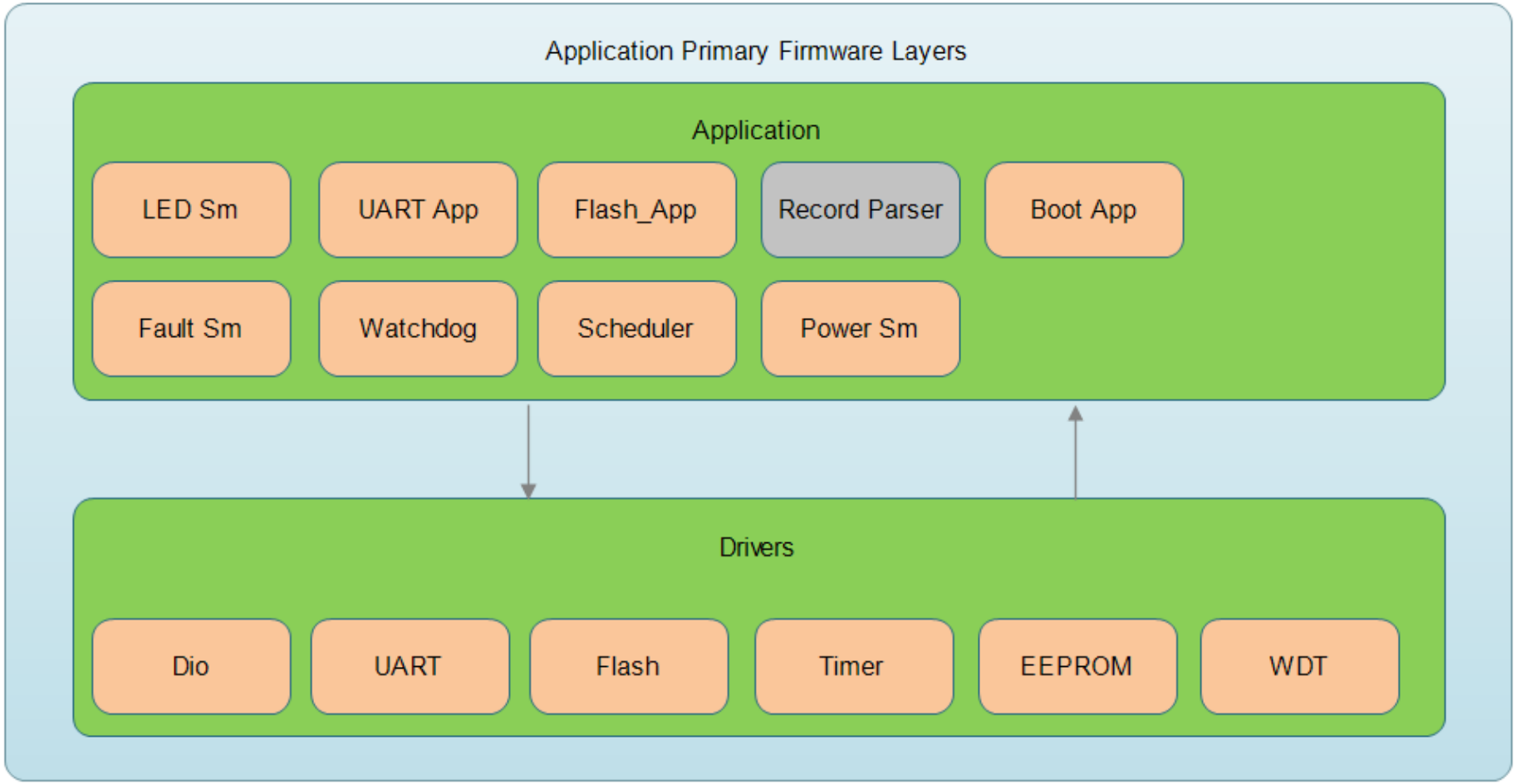
Bootloader Development

- What is needed for a test application?

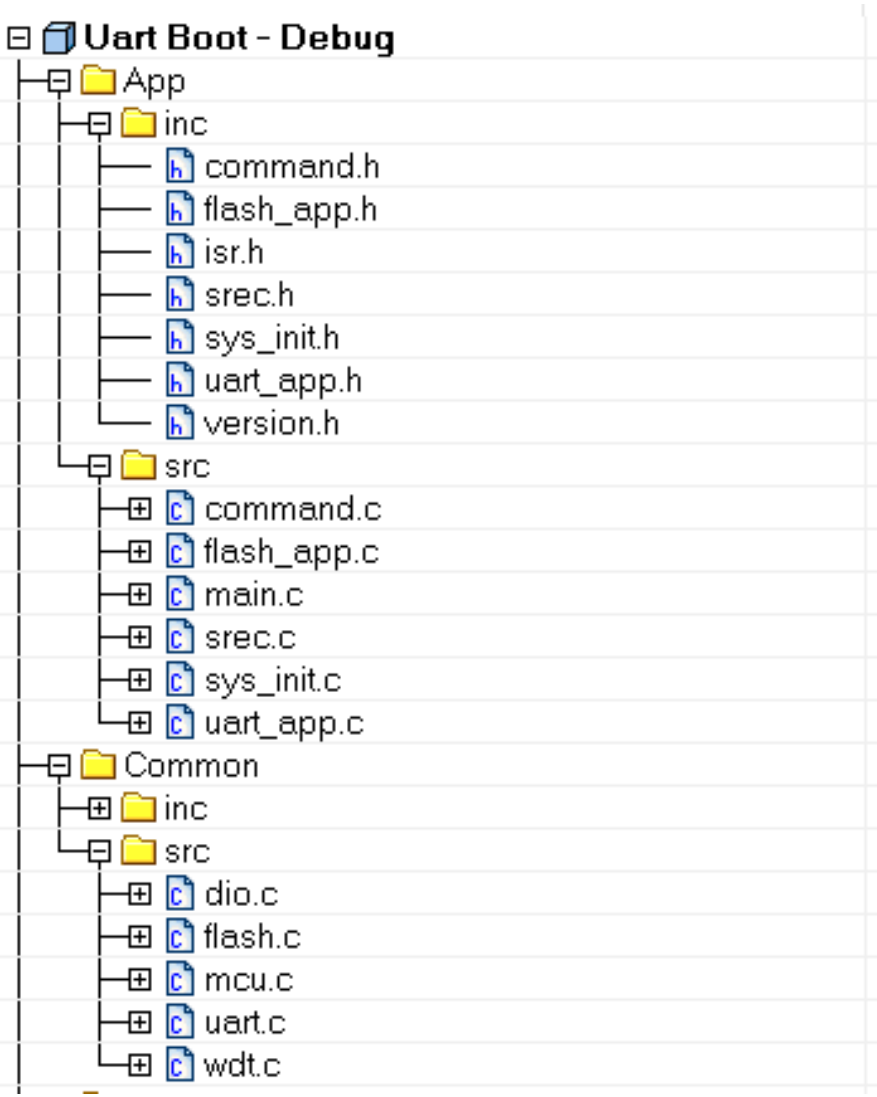
Bootloader Components



Application Components



Project Organization



API's

- Purpose – simplify application programming by abstracting the application into black boxes.
- Critical to creating reusable software
- Defines a common interface that can be used from one project to the next

Creating a HAL

- Steps to develop a HAL for the peripheral
  - Review the microcontroller peripheral
  - Identify peripheral features
    - Identify common MCU elements
    - Identify non-standard MCU elements
  - Design and Create the API Interface
  - Create stubs and documentation templates
  - Implement for Target processor(s)
  - Test
  - Repeat for next peripheral

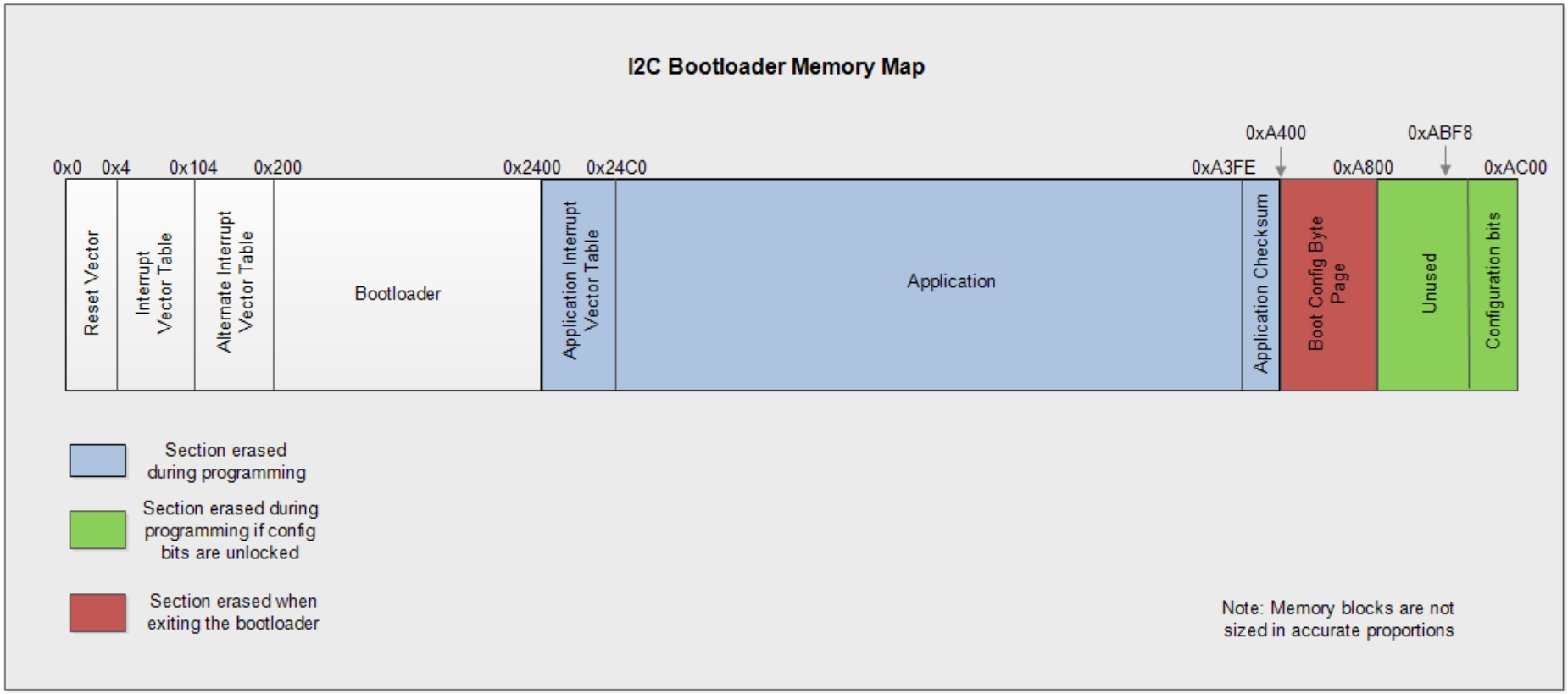
Example HAL Interface

```
void Dio_Init(const Dio_Config_t *Config);
PinLevelEnum_t Dio_ChannelRead(DioChannel_t Channel);
void Dio_ChannelWrite(DioChannel_t Channel, PinLevelEnum_t State);
void Dio_ChannelToggle(DioChannel_t Channel);
void Dio_ChannelModeSet(DioChannel_t Channel, PinModeEnum_t Mode);
void Dio_FunctionModeSet(DioChannel_t Channel, Dio_FunctionSelect_t Func);
```

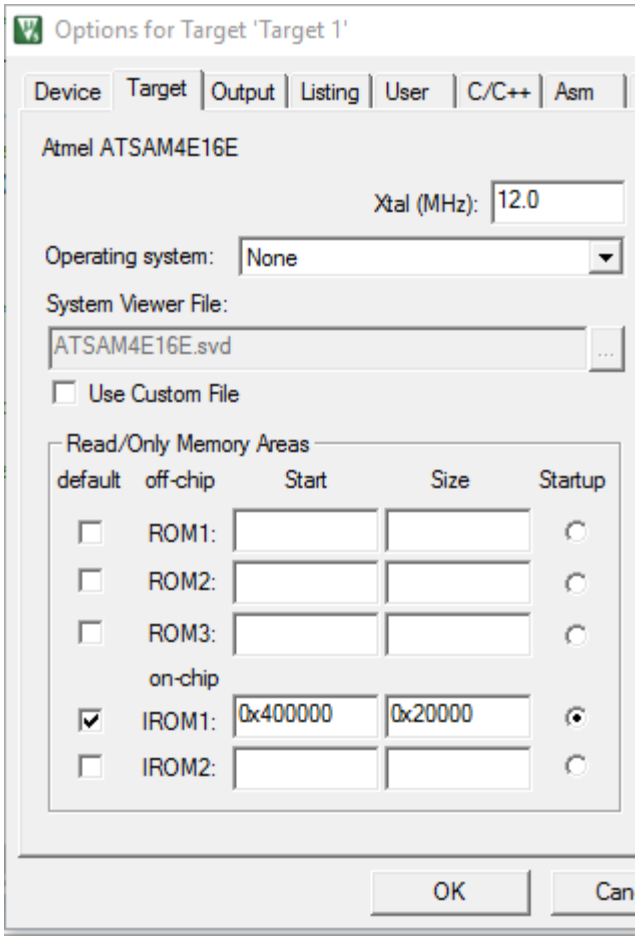
```
void Tmr_Init(const Tmr_Config_t *Config);
void Tmr_Enable(Tmr_Register_t Channel);
void Tmr_Disable(Tmr_Register_t Channel);
```

```
void Wdt_Enable(void);
void Wdt_Disable(void);
void Wdt_Reset(void);
void Wdt_Clear(void);
```

Setting up the linker



Setting up the linker




```
MEMORY
{
  data  (a!xr) : ORIGIN = 0x800,         LENGTH = 0x1FFF
  reset        : ORIGIN = 0x0,           LENGTH = 0x4
  ivt          : ORIGIN = 0x4,           LENGTH = 0xFC
  aivt         : ORIGIN = 0x104,         LENGTH = 0xFC
  program (xr) : ORIGIN = 0x200,         LENGTH = 0x2200
  app_ivt      : ORIGIN = 0x2400,        LENGTH = 0xC0
  checksum     : ORIGIN = 0xA700,    LENGTH = 0x4
  CONFIG4      : ORIGIN = 0xABF8,        LENGTH = 0x2
  CONFIG3      : ORIGIN = 0xABFA,        LENGTH = 0x2
  CONFIG2      : ORIGIN = 0xABFC,        LENGTH = 0x2
  CONFIG1      : ORIGIN = 0xABFE,        LENGTH = 0x2
}

__CONFIG4 = 0xABF8;
__CONFIG3 = 0xABFA;
__CONFIG2 = 0xABFC;
__CONFIG1 = 0xABFE;

__NO_HANDLES = 1;          /* Suppress handles on this device  */

__IVT_BASE  = 0x4;
__AIVT_BASE = 0x104;
__DATA_BASE = 0x800;
__CODE_BASE = 0x200;

__APP_IVT_BASE = 0x2400;
```

Accessing Flash

- What mode?
  - Bit
  - Page
  - sector
- What should the API look like?

```
void Flash_Init(const Flash_Config_t * Config);
void Flash_Write(uint32_t Address, const uint16_t * Data, uint8_t Size);
void Flash_Read(uint32_t Address, const uint16_t * Data, uint8_t Size);
void Flash_Erase(uint32_t Address);
void Flash_RegisterWrite(uint32_t Address, uint32_t Value);
uint32_t Flash_RegisterRead(uint32_t Address);
void Flash_CallbackRegister(FlashCallback_t Function, TYPE (*CallbackFunction)(type));
```

Accessing Flash

```
void Flash_Write(void)
{
    unsigned int i = 0;
    DWORD_VAL Address;

    NVMCON = 0x4003;

    while(BufferedDataIndex > 0)
    {
        Address.Val = ProgrammedPointer - BufferedDataIndex;
        TBLPAG = Address.word.HW;

        __builtin_tblwtl(Address.word.LW, ProgrammingBuffer[i]);
        __builtin_tblwth(Address.word.LW, ProgrammingBuffer[i + 1]);
        i = i + 2;

        asm("DISI #16");
        __builtin_write_NVM();
        while(NVMCONbits.WR == 1){}

        BufferedDataIndex = BufferedDataIndex - 2;
    }

    NVMCONbits.WREN = 0;

}
```

Accessing Flash

```
_Bool SuccessFlag = 0;

/* Verify that the request address space is valid */
if((Address >= BOOTLOADER_CONFIG_START) &&
   (Address <= BOOTLOADER_CONFIG_END))
{
    SuccessFlag = Flash_Write64KbSectorPage(Address, Data);
    //SuccessFlag = Flash_Write8KbSectorPage(Address, Data);
}
else if((Address >= BOOTLOADER_APP_START) &&
        (Address <= BOOTLOADER_APP_END))
{
    SuccessFlag = Flash_Write64KbSectorPage(Address, Data);
}

return SuccessFlag;
```

Command Parsing

```
/**
 *  Defines the command structure used to parse a command packet
 */
typedef struct
{
   uint8_t Command;                          /**< Stores the command */
   void (*function)(uint8_t * Data);         /**< Function pointer to execute on command */
}Command_RxCmdListType;
```

```
/**
 * Defines an array of all of the supported commands and the function that should
 * be executed when the command is received.
 */
const Command_RxCmdListType CommandsList[] =
{
   {BOOT_EXIT,             Command_Exit            },
   {ERASE_DEVICE,          Command_Erase           },
   {PROGRAM_DEVICE,        Command_Program         },
   {QUERY_DEVICE,          Command_Query           },
   {END_OF_COMMANDS,       0x00                    },
};
```

Command Parsing

```
void Command_Process(UartMessageType Message)
{
    const Command_RxCmdListType * CmdListPtr;

    // Loop through the command list and see if there is a match.
    // It will loop until a null pointer is found in the table.
    for(CmdListPtr = CommandsList; CmdListPtr->function != 0; CmdListPtr++)
    {
       // If we find the command we received then exit the loop and execute the command.
       if(CmdListPtr->Command == Message.OpCode)
       {
          break;
       }
    }

    // Do one final check to make sure that a null pointer isn't being executed
    if(CmdListPtr->function != 0)
    {
       // Execute the command and parse out the command byte
       (*CmdListPtr->function)(Message.Data);
    }
}
```

Handling the Watchdog

- Must be enabled
  - Why?
- Integrate into the application
- May be windowed
- Set period for the application appropriately

Resetting the System

- How to reset the system
  - Watchdog timer
    - Infinite loop
    - Illegal write to register
    - Soft reset command
  - Manual software reset
  - Notify user to power cycle

```
void Wdt_Reset(void)
{
  /* Enter an invalid key to force reset */
  SWT.SR.R = 0x0000FFFF;
}
```

Additional Resources

- Download Course Material for
  - Updated C Doxygen Templates (Sept 2015)
  - Example source code
  - Bootloader White Paper
  - Templates
- Microcontroller API Standard
- EDN Embedded Basics Articles
- Embedded Bytes Newsletter

From www.beningo.com under

- Blog and Articles > Software Techniques > CEC Bootloader Design for MCUs

Presented by:

DesignNews

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

Jacob Beningo
Principal Consultant

P.O. Box 400
Linden, Michigan 48451

www.beningo.com

E : jacob@beningo.com
T : 810-844-1522
: Jacob_Beningo
: Beningo Engineering
: JacobBeningo
EDN : Embedded Basics

IEEE computer society Certified Software Development Professional

ARM Connected Community

Newsletters

- Embedded Bytes

http://bit.ly/1BAHYXm

Training

- MicroPython
- Bootloaders
- Low Power Design
- Resume Workshop
- Embedded Techniques

Presented by:

DesignNews

CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS