CDA 3631C Embedded Operating Systems
LAB ASSIGNMENT 5: Programming and Configuring Advanced Peripherals: LCD Display and Joystick

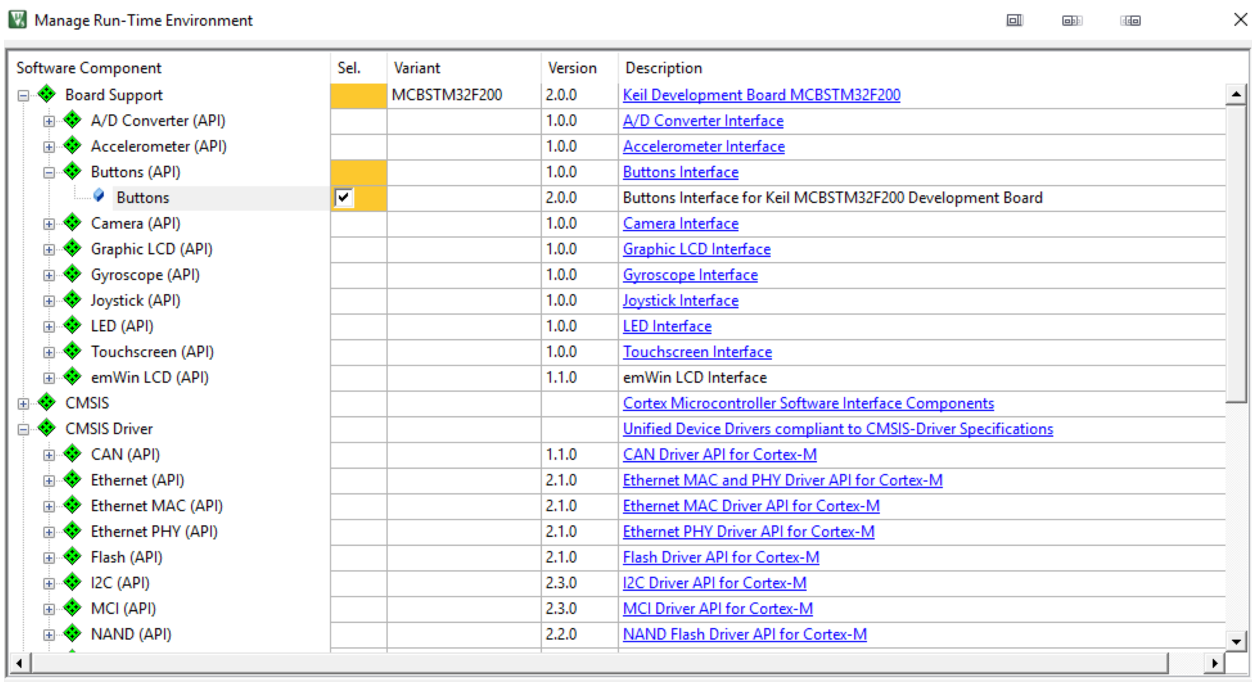
Date Posted: 11 November 2019 Date Due: 22 November 2019 (11:59 PM)

Concept:

- Use the Keil library functions to “easily” access the onboard components.
- Use standalone drivers to access the onboard components.

Introduction:

The developers of the Keil MCBSTM32F200 board wrote several library files to make accessing the extra devices “easier” for the software developer. In uVision5, these libraries may be added to the project in the Manage Run-Time Environment Window. The window with the Buttons library selected is shown below as an example.


Manage Run-Time Environment

Software Component	Sel.	Variant	Version	Description
Board Support	🟨	MCBSTM32F200	2.0.0	Keil Development Board MCBSTM32F200
A/D Converter (API)			1.0.0	A/D Converter Interface
Accelerometer (API)			1.0.0	Accelerometer Interface
Buttons (API)	🟨		1.0.0	Buttons Interface
Buttons	✓		2.0.0	Buttons Interface for Keil MCBSTM32F200 Development Board
Camera (API)			1.0.0	Camera Interface
Graphic LCD (API)			1.0.0	Graphic LCD Interface
Gyroscope (API)			1.0.0	Gyroscope Interface
Joystick (API)			1.0.0	Joystick Interface
LED (API)			1.0.0	LED Interface
Touchscreen (API)			1.0.0	Touchscreen Interface
emWin LCD (API)			1.1.0	emWin LCD Interface
CMSIS				Cortex Microcontroller Software Interface Components
CMSIS Driver				Unified Device Drivers compliant to CMSIS-Driver Specifications
CAN (API)			1.1.0	CAN Driver API for Cortex-M
Ethernet (API)			2.1.0	Ethernet MAC and PHY Driver API for Cortex-M
Ethernet MAC (API)			2.1.0	Ethernet MAC Driver API for Cortex-M
Ethernet PHY (API)			2.1.0	Ethernet PHY Driver API for Cortex-M
Flash (API)			2.1.0	Flash Driver API for Cortex-M
I2C (API)			2.3.0	I2C Driver API for Cortex-M
MCI (API)			2.3.0	MCI Driver API for Cortex-M
NAND (API)			2.2.0	NAND Flash Driver API for Cortex-M

To use a given library, you must select it in the Manage Run-Time Environment (RTE)window. This will automatically add the C library file to your project (and it will be locked so you can’t accidentally alter it). You must also “#include” the header file (*filename.h*) in any C file that uses the library’s functions. The name of the header file is not explicitly stated in the window, but it may usually be found by opening the corresponding C library file that was added to the project (since code files almost always include their own headers).

Although this driver management interface is meant to be helpful, it masks a lot of what is happening behind the scenes and can make for a very steep learning curve. The library developersalso had to implement a hardware abstraction layer (HAL) to make some of the component libraries function. This means that one, significant code is added to the project when using very simple components, and two, that some processor components have been “claimed” by the HAL and cannot be used directly by the application.

To simplify the first attempt at using the evaluation board, the recommended approach will use a few of the supplied libraries added through the RTE and a few stand-alone drivers. For a stand-alone driver, the header file and source code file are managed directly by the programmer. Both files need to be in a directory that the compiler will search (for simplicity, they may be placed in the same directory as the regular source code). The header file must be “#include”ed in any C file that uses the functions, and the source code file must be manually added to the project.

Assignment:

First save Serial.c, Serial.h, I2C.h, I2C_stm32fxx.c, Joy.c, and Joy.h in your Project directory folder. Create a new project and write a program for the Keil MCBSTM32F200 board that meets the following requirements:

1. The LCD screen must display the printable ASCII characters in a 6 row by 16 column arrays formatted as shown. The text must use the 16x24 font and be white with a purple background, and the rest of the display must be purple. The text must be centered on the LCD screen (i.e. leave two columns on both sides and skip down two rows.)

	!	“	#	$	%	&	‘	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	O
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Note that each character has a hexadecimal ASCII value, and the first row all has the 0x20’s in order from 0x20 to 0x2F, the second row has 0x30 to 0x3F, and so on.

2.When the system initializes, the “space” character should be highlighted.

3. When the board highlights a character, it must also write the ASCII value to the LEDs with the most significant bit of the character to the **left** and the least significant to the **right** (i.e. as one would normally write the bits of a binary value left to right).

4. Use the SysTick interrupt to sample the joystick ~10 times a second. When a change in the joystick position is found, the cursor/highlighted letter must move one position in the direction indicated. The cursor must not move outside the bounds of the 6 rows by 16 column arrays.

5. Your code must be usefully commented.

Helpful Advice:

- The first line of your C code is often "SystemCoreClockUpdate()", which uses the clock configuration used in the automatically generated startup code to store the processor's clock frequency in "SystemCoreClock", which is used by a lot of initialization functions.

- Use the Manage RTE interface to add the LCD screen (Board Support->Graphic LCD) and the LEDs (Board Support->LED). The CMSIS->Core and Device->Startup were also needed. Each board component needs you to call its initialization function in the main program before it can be used.
-

  • When using the Keil files, you will need to use their HAL. For this project, this can be done by "#include"ing the "stm32f2xx_hal.h" file, and calling HAL_Init() in the main program. Note that this will also completely initialize the Systick device to periodically interrupt, so all you need to do is write the Systick_Handler ISR.

- Use the stand-alone files to add the joystick (JOY), the USART3 serial port (serial), and the I2C communication port (I2C), which is used to communicate with the GLCD and is not used directly by your code. Each board component needs you to call its initialization function in the main program before it can be used.

- The LEDs display values with the most significant bit to the right, not the left as needed. You will need to reverse the bits.

- Follow the Template Provided and write the code for the corresponding Functions as per the descriptions provided in comment. Systick_handler code is already provided, System Initialization function **system_setup()** is already provided.

Deliverables:

1) Entire project in Zip file.
2) Report with cover page, Assignment Description, Screenshot of your Board LCD showing all ASCII characters, detailed conclusion