ตัวอย่างการพัฒนาโปรแกรม ด้วย **JTAGICE mkll**

 สั่ง Run Program AVR Studio4 แล้วเลือกสร้าง project ใหม่ โดยเลือกที่ project → New project จากนั้นเลือกกำหนดตัวเลือกต่างๆให้กับโปรแกรมโดยให้เลือก Project type เป็น AVR GCC ส่วน Location สำหรับบันทึก project ให้ระบุตำแหน่ง Folder ที่ต้องการใช้บันทึก ไฟล์ และ Code ต่างของ project และกำหนดชื่อของ Poject name ตามต้องการในที่นี้ให้ ลองกำหนดเป็น demo_jtagice_mkll และ ให้เลือก Create initial file ไว้ด้วย ดังนี้

Atmel AVE Assembler	Create folder
Create initial file Initial file: demo_itagice_mkll	Create folder
demo_ttagice_mkll	.c
Location C.\ET_BASE_AVR\ET-BASE xMEGA\W/mAVR\DEMD	
3.700	

2. เมื่อกำหนดค่าตัวเลือกต่างๆ ให้กับโปรแกรมเรียบร้อยแล้ว ให้เลือกที่ Next แล้วกำหนดค่าใน Debug platform เป็น AVR Dragon และเลือก Device เป็น ATxmega128A1 ซึ่งเมื่อสร้าง project เสร็จโปรแกรมจะสร้างไฟล์ภาษาซีให้ โดยมีชื่อเดียวกับ project ไฟล์ ซึ่งในที่นี้จะเป็น ไฟล์ชื่อ demo_jtagice_mkll.c ให้เองโดยอัตโนมัติ เพียงแต่ไฟล์ดังกล่าวจะยังไม่มี code ใดๆ บรรจุไว้ให้ เป็นเพียงหน้ากระดาษเปล่าๆ ซึ่งต้องรอให้เราเขียน code เพิ่มเข้าไปเอง ดังรูป



3. ให้พิมพ์คำสั่งของโปรแกรมสำหรับทดสอบการทำงาน ในหน้าต่าง Text Editor ของโปรแกรม โดยในที่นี้จะทดสอบด้วย Codeโปรแกรม สำหรับทำหน้าที่ทดสอบการทำงานของบอร์ดใน เบื้องต้น โดยทำหน้าที่ ON/OFF LED ซึ่งต่อควบคุมจากขา PO2 โดยโปรแกรมตัวอย่างจะยัง ไม่มีส่วนของการหน่วงเวลาเข้ามาเกี่ยวข้อง ซึ่งเมื่อต้องการ Run ด้วยความเร็วจริงต้องเพิ่ม คำสั่งเกี่ยวกับการหน่วงเวลาแทรกเข้าไปในภายหลังด้วย ดังตัวอย่าง

//PQ2 = Out

//PQ2 = High

//PQ2 = Low

```
#include <avr/io.h>
int main(void)
{
    PORTQ.DIRSET = (1<<2);
    while(1)
    {
        PORTQ.OUTSET = (1<<2);
        PORTQ.OUTCLR = (1<<2);
    }
}</pre>
```

 หลังจากพิมพ์ Code โปรแกรมเสร็จแล้วให้สั่งแปลโปรแกรม โดยเลือกที่ build →rebuild all ซึ่งถ้าทุกอย่างถูกต้อง ผลการแปลคำสั่งจะได้ผลลัพธ์เป็น "Build succeeded with 0 Warnings... " และจะรายงานผลการแปลพร้อมขนาดหน่วยความจำที่ใช้ไป และจะได้ Output เป็น HEX File ที่มีชื่อเดียวกันกับ project ที่สร้างไว้ โดยจะบรรจุอยู่ในDirectory ย่อย ชื่อ default ดังรูป



5. ต่อสาย USB ให้กับ JTAGICE mkll พร้อมกับต่อสายสัญญาณ JTAG จาก JTAGICE mkll เข้ากับขั้ว JTAG ของบอร์ด ET-BASE xMEGA128A1 ดังรูป



6. สั่ง Start Debug ซึ่งโปรแกรมจะเปลี่ยนเข้าไปทำงานใน Debug Mode ทันที จากนั้นให้ลอง ทดสอบ สั่งงาน Debug ต่างๆดู เช่น เลือกทดสอบแบบ Auto Step แล้วดูผลการทำงานซึ่งจะ เห็นมีลูกศรสีเหลืองชี้หน้าบรรทัดคำสั่ง ที่โปรแกรมกำลังทำงานอยู่ในขณะนั้น พร้อมกับเห็น การเปลี่ยนแปลงค่ารีจิสเตอร์ต่างๆ และที่บอร์ดเองก็จะเห็น LED กระพริบตามการทำงานของ โปรแกรมด้วยเช่นเดียวกัน ซึ่งจะเห็นว่าการทำงานต่างๆของโปรแกรมเหมือนจริงทุกประการ ยกเว้นเรื่องความเร็ว

การพัฒนาโปรแกรมของบอร์ด ET-BASE xMEGA128A1 ด้วย JTAGICE mkll



ซึ่งผู้ใช้สามารถทดสอบการทำงานของโปรแกรมผ่านความสามารถในการ Debug ได้มากมาย หลายแบบ ไม่ว่าจะเป็นการตั้ง Break เพื่อหยุดการทำงานในบรรทัดของโปรแกรมที่สงสัยเพื่อ ตรวจสอบดูผลการทำงานต่างๆ ณ. จุดนั้นว่า ถูกต้องตามที่ออกแบบโปรแกรมไว้หรือไม่ ซึ่งเทคนิค วิธีต่างๆเกี่ยวกับการใช้งาน Debug ของ JTAGICE mkll สามารถศึกษาเพิ่มเติมได้จากคู่มือการใช้ งานของ JTAGICE mkll เอง

การพัฒนาโปรแกรมของบอร์ด ET-BASE xMEGA128A1 ด้วย JTAGICE mkll

นอกจากจะใช้ JTAGICE mkll ในการทำงานแบบ Debug เพื่อทดสอบการทำงานหรือ ตรวจสอบความผิดพลาดของโปรแกรมแล้ว เรายังสามารถใช้ JTAGICE mkll ทำการโปรแกรม Hex ให้กับ MCU ภายในบอร์ดเพื่อนำไปใช้งานจริงได้ด้วย โดยโปรแกรมที่จะใช้ Run จริง จะต้องแก้ไข เพิ่มเติมให้สมบูรณ์ครบถ้วน เช่นจากโปรแกรมตัวอย่างการทดลองที่ผ่านมาก็จะต้องมีการเพิ่มส่วนของ โปรแกรมหน่วงเวลาเข้าไปด้วย ไม่เช่นนั้นเราจะมองไม่เห็นการติด ดับ ของ LED เนื่องจากในขณะ ทำงานจริงๆ MCU จะทำงานเร็วมากดังนั้นจึงต้องมีการเพิ่มเติมส่วนของโปรแกรมหน่วงเวลาเข้าไป ด้วยดังตัวอย่าง

```
#include <avr/io.h>
#define F_CPU 200000UL
#include <util/delay.h>
int main(void)
{
    PORTQ.DIRSET = (1<<2); //PQ2 = Out
    while(1)
    {
        PORTQ.OUTSET = (1<<2); //PQ2 = High
        _delay_ms(250);
        PORTQ.OUTCLR = (1<<2); //PQ2 = Low
        _delay_ms(250);
    }
}</pre>
```

เมื่อทำการแก้ไขโปรแกรมเรียบร้อยแล้วให้ สั่งแปลโปรแกรมซ้ำใหม่อีกครั้ง โดยเลือกที่เมนู คำสั่ง Build —> Rebuild ALL

จากนั้นให้เลือกที่เมนู Tools → Program AVR → Connect และในส่วนของ Select AVR Programmer ให้เลือก JTAGICE mkII → USB → Connect ซึ่ง JTAGICE mkII สามารถที่จะ เชื่อมต่อกับบอร์ด ET-BASE xMEGA128A1 ในโหมด Programming ได้ 2 แบบ คือ เชื่อมต่อผ่านทาง ขั้ว JTAG mode ดังวิธีการข้างต้นที่ผ่านมา หรือ เชื่อมต่อผ่านทาง PDI mode ซึ่งทั้ง 2 แบบสามารถ ทำงานได้เหมือนกัน แต่ในกรณีที่จะใช้ PDI mode จะต้องใช้สายแพร์ Convert เพื่อเชื่อมต่อลัญญาณ ด้วย เนื่องจาก JTAGICE mkII ไม่ได้ทำขั้วแบบ PDI 6 Pin ไว้ให้โดยตรง โดยใช้สายสีแดง(9) เป็น PDI Data,สายสีเขียว(6)เป็น PDI Clock(RESET),สายสีม่วง(4)เป็น +VDD และ สายสีขาว(2)เป็น GND แต่เพื่อความสะดวกในการใช้งานแนะนำให้ใช้การเชื่อมต่อแบบ JTAG mode จะดีกว่า

การพัฒนาโปรแกรมของบอร์ด ET-BASE xMEGA128A1 ด้วย JTAGICE mkll

▶ AVR Studio - [CAET_BASE_AVRATEBASE xAREGAWinAVR0EMDAdemo_tapico_mkil.c] この意見のまたまた。 この意見のまたまた。 このでは、このでは、このでは、このでは、このでは、このでは、このでは、このでのでは、また。	11日のの <i>と</i> 、2日日の日	
E File Project Build Edit View Tools Debug Window Help		- 5×
AVE GCC H Build F7 avr./io.h>	~	I/O View ¥
come_ragee Rebuild All Crite 200000001 Source Rebuild All Crite 200000001		25 - (E E
G demo E Build and Rup Ctri-F7		Name V 🔿
Alt+F7		Analog Comparator A
B		📧 🏶 Analog Comparator B
Cher File Export Make le		Analog to Digital Conv
		Analog to Digital Conv
delay ns(250); //PQ2 = High		ALS Lippto Module
FORTQ OUTCLE = (1<<2); //PQ2 = Low		Advanced waveform
_delay_ns(250);		Advanced waveloini
		CPU Begisters
		# Digital to Analog Conv
1		🕣 🛡 Digital to Analog Conv
6		DFLL for 2MHz RC 0s
		A DFLL for 32MHz RC 0
		DMA Controller
		External Bus Interface
		Event System
i l		General Purpose IU R
6 I		High Parch tion Exten
5		High-Besolution Exten
5 I I I I I I I I I I I I I I I I I I I		# High-Besolution Exten
2		🗃 🗣 IR Communication Mo 🤜
		<
1 II.		
6		Name Address \
5 I I I I I I I I I I I I I I I I I I I		
1		
1 II		
6 1		
5 1	9	
	8	
C:\ET_BASE_AVR\ET-BASE xMEGA\WinAVR\DEMO\demo_jtagice_mkILc	4 Þ	
	* ×	
Program. 528 bures (0.55 mill)	<u>^</u>	
(.text + .dat + .houtlader)		
Data: 0 bytes (0.0% Pull)		
(.data + .bss + .noinit)		
Build succeeded with 0 Warnings		
🖪 Build 🚯 Message 🖼 Find in Files 📴 Breakpoints and Tracepoints		< 11 ×
	ATxmega128A1 JTAGICE mkll Auto Running 💮	Ln 8, Col 1 CAP NUM OVR

 \downarrow



เมื่อเข้าสู่หน้าต่างของ JTAGICE mkll ใน Tab ของ Main ให้เลือกกำหนดเบอร์ของ MCU เป็น ATxmega128A1 แล้วเลือก Programming Mode and Target Settings เป็น JTAG mode ถ้า เชื่อมต่อแบบ JTAG หรือ เลือกเป็น PDI ถ้าเชื่อมต่อแบบ PDI จากนั้นให้ทดสอบ Read Signature ของ MCU ดูว่าสามารถอ่านได้ถูกต้องหรือไม่ เมื่อเรียบร้อยแล้วให้เปลี่ยนไปที่ Tab ของ Program แล้ว เลือก Erase device before falsh programming และ Verify device after programming แล้วเลือก กำหนด Hex File ที่ได้จากการแปลคำสั่งในโปรแกรม แล้วเลือก Program ดังรูป

JTAGICE mkll in JTAG mode with ATxmega12	2841
Main Program Fuses LockBits Advanced HV Device and Signature Bytes	V Settings HW Info Auto Erase Device Read Signature
Programming Mode and Target Settings	JTAGICE mkli in JTAG mode with ATxmega128A1 Image: Constraint of the settings Main Program Fuses LockBits Advanced HW Settings Device Image: Constraint of the settings Erase Erase Erase Erase Application Image: Constraint of the settings Image: Constraint of the settings
Setting device parameters DK!	Flash Input HEX File C:\ET_BASE_AVR\ET-BASE xMEGA\WinAVR\DEMO\c Program Verify Read EEPROM Use Current Simulator/Emulator EEPROM Memory Input HEX File C:\ET_BASE_AVR\AVR_ISP_MKI\Projects\AVRISP-MK
Entering programming mode DK1 Reading signature 0h1E, Ux97, 0x4C DK1 Leaving programming mode DK1	ELF Production File Format Input ELF File: Save From: FLASH FUSES LOCKBITS Fuses and lockbits settings must be specified before saving to ELF Fragram File Save Save Save Save Saving to ELF

ซึ่งในกรณีที่ใช้โปรแกรม AVR Studio Version 4.18 build 700 ร่วมกับ JTAGICE mkll ใน โหมด programming นี้พบว่ายังมีข้อผิดพลาดอยู่ คือ เมื่อสั่ง Program เสร็จเรียบร้อยแล้ว MCU จะ ยังคงค้างการทำงานอยู่ใน JTAG Mode อยู่ ถึงแม้ว่าจะกดรีเซ็ตแล้วก็ตาม MCU ก็จะยังคงไม่ทำงาน ซึ่งต้องแก้ปัญหาโดยการสาย JTAG ออกจากขั้ว JTAG ของบอร์ดแล้วทำการ Power-Up ใหม่ก่อน ด้วย 1 ครั้ง MCU จึงจะสามารถเริ่มทำงานตามโปรแกรมที่ได้สั่งโปรแกรมไว้แล้ว