הנחיות לשילוב קוד עדכון בקוד הכוונת של BL

# טרמינולוגיה של סוגי הקוד

## System Memory

אזור ROM בזיכרון שבו ST כתבו

1. את הקוד שמתחיל את האפליקציה מבנק 2 במידה והביט nBFB2 = 0 2
2. הקוד Bootloader של ST (להלן STBL)

## קוד עדכון

קוד שנכתב ב-C שמטרתו לנהל את הכניסה לקוד ״הרגיל״

הוא כולל את:

* הקבצים: BL.c, BL.h
* הפונקציה:  
   DM5\_STATUS\_ENUM EnterLoaderMode(int unit,ICD\_MSG\_TYPE \*pAanlyzedInputMsg)
* שדרוג של הפונקציה: void SystemInit (void)

לשם פשטות אחזקת הקוד, קוד העדכון נכתב כך שהוא יהיה זהה בקוד המפעלי ובקוד המתעדכן.

## הקוד ״הרגיל״

שאר הקוד   
בקוד ״הרגיל״, ההבדל בין הקוד המפעלי לקוד המתעדכן הוא רק בכתובת ההתחלה של ICONS\_START\_ADDRESS

## קוד מפעלי

קוד עדכון + הקוד הרגיל שנצרב במפעל (צהוב בטבלה שבפסקה FLASH שלהלן)

## קוד מתעדכן

קוד עדכון + הקוד הרגיל שניתן לצרוב בעדכון גרסה בשטח (אפור בטבלה שבפסקה FLASH שלהלן)

## קוד האפליקציה

הקוד שקב״מ יכתוב

# שינויים נדרשים בקוד הרגיל

להלן רשימת השינויים והתוספות לקוד הכוונת, על מנת שהוא יכיל גם את קוד העדכון:

## Include

בקובץ שמכיל אל הפונקציה EnterLoaderMode ובקובץ צריך להוסיף

#include "BL.h"

ולכלול בפרוייקט את הקבצים BL.c ; BL.h מהקוד עדכון

## EnterLoaderMode

יש להוסיף לפוקנציה EnterLoaderMode (אשר נמצאת בקוד הכוונת) הפנייה לפונקציה *GoToBootLoader* (אשר נמצאת בקוד העדכון) כדלקמן:

DM5\_STATUS\_ENUM EnterLoaderMode(int unit,ICD\_MSG\_TYPE \*pAanlyzedInputMsg){

  DM5SysState = SYS\_STATE\_LOADER;

  Build\_DM5\_Status();

*// The following command comprise the change in* EnterLoaderMode

*GoToBootLoader(); // GoToBootLoader is implemented in BL.c*

  return DM5\_OK;}

## SystemInit

בעיקרון אפשר להשתמש בקובץ system\_stm32l1xx.c שבקוד עדכון, אבל זה לא נבדק.

יותר בטוח להעתיק את החלקים הרלוונטים.

בפרט:

להחליף בקוד הכוונת את קטע הקוד:

void SystemInit (void(

}

בקטע הקוד הבא:

#include "BL.h"  
#define BootLoaderMemoryLocation 0x1FF00000

void SystemInit (void(  
}  
… Omitted code lines, should be copied from system\_stm32l1xx.c …

if(OB->USER & (FLASH\_OBR\_nRST\_BFB2 >> 16U))  
{// MCU was Set to be boot from bank 1   
… Omitted code lines, should be copied from system\_stm32l1xx.c … SysMemBootJump();}

else{ // Running Code from bank 1

SCB->VTOR = FLASH\_BASE | VECT\_TAB\_OFFSET;

}

else // run from bank2

{

SCB->VTOR = FLASH\_BANK2\_BASE | VECT\_TAB\_OFFSET;

}

וצריך למחוק את הקינפוג המקורי של SCB->VTOR

#ifdef VECT\_TAB\_SRAM

SCB->VTOR = …

#else

SCB->VTOR = …

#endif

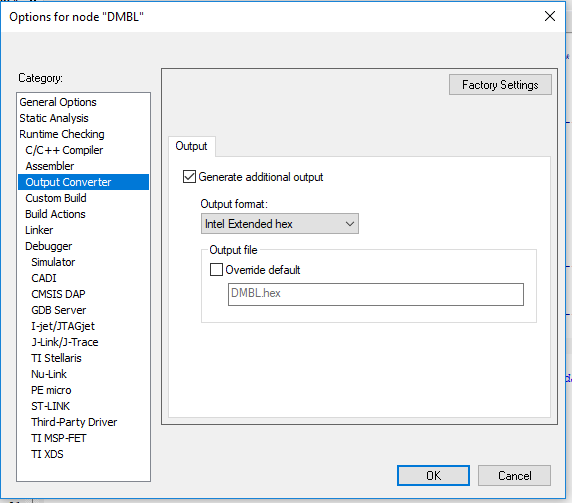
{

## Linker file

בעת צריבת הקובץ המפעלי   
ובעת הכנת (make) הקובץ המתעדכן  
צריך לקנפג את כתובות הזיכרון שאליהם הקוד נכתב בהתאם להוראות ב-FLASH / השוני המובנה בין הקוד המפעלי לקוד המתעדכן שלהלן.

דרך אגב, צריבה נעשית כמו בכל התוכנות שבכל הכוונות

והכנת קובץ עם קוד מתעדכן נעשית על ידי קינפוג output converter ב-IAR (כמו בתמונה שלהלן)



# זיכרון

## EEPROM

הקוד עדכון משתמש בכתובת

#define EEPROM\_Address\_Mode\_Of\_Execution 0x08081FF0

כלומר קוד הכוונת צריך לא להשתמש בכתובת הזאת (או שפשוט לשנות את הערך ב-define)

## FLASH / השוני המובנה בין הקוד המפעלי לקוד המתעדכן

השוני המובנה היחיד בין הקוד המפעלי לקוד המתעדכן הוא במיקומם בזיכרון.  
קוד העדכון זהה בשניהם.  
קוד הכוונת מן הסתם צפוי להתעדכן בגרסאות מאוחרות בקוד המתעדכן.  
קוד הכוונת עשוי להתעדכן בקוד המפעלי, אבל רק בעת מכירה (בטרם המסירה ללקוח).

לאזור Flash של הזיכרון בדר"כ צורבים, ובמקרה שלנו גם מעדכנים את התוכנה של הרכיב  
חלוקת הכתובות בתוך הזיכרון מתוארת בטבלה שלהלן

|  |  |
| --- | --- |
| קוד מפעלי  Factory Code @ Bank1  Start address 0x0800 0000  End address 0x0800 7FFF | קוד מתעדכן  Updated Code @ Bank2 Start address 0x0804 0000  End address 0x0804 7FFF |
| קוד מתעדכן  Updated Reticles & Menus @ bank 1  Start address 0x0800 8000  End address 0x0803 FFFF | קוד מפעלי  Factory Reticles & Menus @ bank2  Start address 0x0804 8000  End address 0x0807 FFFF |

כדי שהחלק של ה- Reticles & Menusאכן ייכתב במקום הנכון, יש להגדירו הן בתוכנה והן בקובץ linker

בתוכנה נדרש להגדיר

בקוד המפעלי:

#define ICONS\_START\_ADDRESS 0x08048000

ובקוד המעתדכן:

#define ICONS\_START\_ADDRESS **0x08008000**

כדי ליצור (make) את הקוד המפעלי, אפשר לקנפג את ה-linker file (stm32l152xe\_flash.icf) באמצעות כלי מובנה ב-IAR, בשם linker configuration file editor

הפרמטרים עבור יצירת קוד מפעלי:

.intvec start: 0x08000000

ROM start: 0x08000000

ROM end: 0x0807FFFF

הפרמטרים עבור יצירת קוד מתעדכן:

.intvec start: 0x08040000

ROM start: **0x08008000**

ROM end: 0x08047FFF

כמו כן, כדי להכין את הקוד המתעדכן בפורמט שמעבירים לאפליסופט, יש לבחור ב-IAR:  
Right Click on project name 🡪 Options 🡪 Output Converter 🡪 Generate additional output 🡪 Raw binary

להלן הסיבות לפיצול של הקוד:

מחד גיסא, אזור הזכרונות הנגיש לצריבה על ידי STBLהינו 0x08000000 - 0x08060000 (כל בנק 1 וחצי מבנק 2), לכן הקוד המתעדכן לא יכול להשתמש בחצי השני של בנק 2.

מאידך גיסא, כדי להפעיל את STBL הקוד שקופץ אל System Memory צריך לרוץ עם קינפוג nBFB2 = 1 , (אחרת System Memory קופץ לבנק 2 מבלי להפעיל את STBL).   
המשמעות של nBFB2 = 1 היא שעם איתחול, הקוד שמותחל הינו הקוד שנמצא בבנק 1.  
ערכו של nBFB2 לא יכול להשתנות בזמן עדכון התוכנה. לכן בעת תקלה במהלך עדכון התוכנה, איתחול יפעיל את הקוד שבבנק 1. ולכן כדאי שבתחילת בנק 1 יהיה הקוד המפעלי שלא מתעדכן.  
כלומר הקוד המתעדכן לא יכול להיות בתחילת בנק 1, בנוסף לכך שהוא לא יכול להיות בסוף בנק 2.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |