



AT03160:ATxmega128A1から他のAtmel XEMGAデバイスへの ブートローダ移植

Atmel AVR XMEGA

要点

- Atmel[®] AVR[®] XMEGA[®]フ゛ートロータ゛
- 手順応用
- 自己プロフラミング用Cコード試供応用
- フラッシュ メモリとEEPROMの両メモリの読み書き
- 施錠ビット読み書き
- ヒューズビット読み込み

説明

多くの電機設計が急速に進化するため、既に出荷または販売された製品を更新できることが 必要になります。マイクロコントローラに組み込まれたブートローダファームウェアは書き込み器の必要なし に応用フラッシュ更新を容易にすることができます。

この応用記述はAtmel ATxmega128A1から他のAVR XMEGAデバイスへブートローダを段階的に 移植する方法を記述します。この応用記述がAVR1605用に拡張された読み物のため、この応 用記述を読む前に、AVR1605:XMEGAブートローダ即時開始の手引きを読むべきです。AVR1605 のプロジェクトはATxmega128A1に基づき、プロジェクトー括はAtmelウェブサイトからタウンロートすること ができます。

自己プログラミングについてのより多くの情報に関してはAVR109:自己プログラミング応用記述を参照してください。

目次

1.	手順 ••••••••••••••••••••••••••••••••••••	• 3
	1.1. 手順1 - IAR開始・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	• 3
	1.2. 手順2 - プロジェクトを開く ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	• 3
	1.3. 手順3 - プロセッサとxclファイルの変更 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	• 4
	1.4. 手順4 - フラッシュ ページ容量変更 ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	• 6
	1.5. 手順5 - 定義情報変更 · · · · · · · · · · · · · · · · · · ·	• 6
	1.6. 手順6 – プロジェクトのコンパイル ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	• 7
	1.7. 手順7 – ブートローダのデバッグ・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	• 8
	1.8. 手順8 – bat(バッチ)ファイルの変更 ······	• 9
2.	推奨読み物・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・	• 9
3.	改訂履歴 ••••••	• 9

1. 手順

ここはATxmega128A1から他のXMEGAデバイスへアートローダを移植するための主な手順です。手続きに従って移植を行う前にATxmeg a128A1に基づく元のプロジェクトがリンクのhttp://www.atmel.com/Images/AVR1605.zipを通してダウンロートされるべきです。

移植手続きを開始する前にIAR Embedded Workbench 6.11版またはそれ以降が最初にインストールされるべきです。 ブートローダがAtmel Studioでプログラミングされる場合、任意選択のAtmel Studio 6.0版またはそれ以降がインストールされます。

1.1. 手順1 - IAR開始

Windowsのスタート⇒プログラム⇒IAR Systems⇒IAR Embedded Workbench for Atmel AVR⇒IAR Embedded Workbenchをクリックしてください。

図1-1. IAR開始	
iAR Embedded Workbench for Atmel AVR 6.11	IAR Embedded Workbench
🛑 IAR Systems License Activation	📝 IAR Product Information Log File
🕜 IAR Systems License Manager	💿 Release Notes

1.2. 手順2 - プロジェクトを開く

File(ファイル)⇒Open(開く)⇒Workspace(作業空間)をクリックしてください。



プロジェクトフォルダを閲覧して"bootloader.eww" プロジェクトファイルを選択し、そして"Open(開く)" 釦をクリックしてください。

図1-3. bootloa	derを開く			
Open Workspa	e			?×
Look in:	🗀 code	G 6	D 📂 🛄 -	
My Recent Documents	AVROSP AVROSP_Test doxygen			
My Documents				
My Computer				
	File name: bootloader.eww			pen
	Files of type: Workspace Files (^{,*} .eww)	Ca	ancel

使ったIARソフトウェアの版がより新しい場合、 変換指示が発生するでしょう。続けるには" Yes"をクリックしてください。

IarldePm Image: Constraint of the project file 'bootloader.ewp' is in an old format. Would you like to convert it for use with this version? (The converted project will not work with older versions of EW, but a backup copy of the original file will be made.) Yes No	図1-4. 3	変換指示
The project file 'bootloader.ewp' is in an old format. Would you like to convert it for use with this version? (The converted project will not work with older versions of EW, but a backup copy of the original file will be made.)	laridePn	n 🛛 🔀
	2	The project file 'bootloader.ewp' is in an old format. Would you like to convert it for use with this version? (The converted project will not work with older versions of EW, but a backup copy of the original file will be made.) Yes No

1.3. 手順3 - プロセッサとxclファイルの変更

作業空間内でbootloader - Debugを選択し、そしてProject(プロジェクト)⇒ Options(任意選択)をクリックしてください。

図1-5. 任意選択(Options)を開く							
🔀 bootloader - IAR Embedded Workbench IDE							
File Edit View	Project Tools Window	Help					
Vorkspace Debug Files Dotto: bootlos b	Add Files Add Group Import File List Edit Configurations Remove Create New Project Add Existing Project						
	Version Control System	Alt+F7					
├── ि serial ├── ि serial └── ि sp_dr └── ि ௺ sp_dr └── ☐ ŵ sp_dr	Make Compile Rebuild All Clean Batch build	F7 Ctrl+F7 F8					

4

Category(区分)でGeneral Optionsそしてその後にTarget(目的対象)タブを選択し、ATxmega128A1を目的対象デバイスに変更してください。

図1-6. プロセッサ変更	
Options for node "bootloader"	
Category	
Assembler	
Custom Build Target Output Library Configuration Library Options Heap Configu	
Build Actions	
Linker Processor configuration	
Debugger Auto outru	
ICE200	ATxmega128A1
JTAGICE Use 64-bit doubles No MUL instruction ATtiny	ATxmega128A1U
JTAGICE3 ✓ Utilize inbuilt EEPROM. Size (no. of bytes): 2048 ATxmega →	ATxmega128A3
JTAGICE mkII Generic Devices	ATxmega120A3U
Dragon Other Other	ATxmega128B1
Third-Party Driver	ATxmega128B3
	ATxmega128D3
System configuration	ATxmega128D4
Configure system using dialogs (not in .XCL file)	ATxmega16A4
FPSLIC partitioning	ATxmega16A4U
	ATxmega16D4
	ATxmega192A3
OK Cancel	ATxmega192A3U
	ATxmega19203

図1-7. xcl7rイル変更

/**************************************
/* Segments in program address space (internal Flash memory) */
/**************************************
-DX_INTVEC_SIZE=1F4 // 4~イト×125~7タ
-DX_APPLICATION_SECTION_START=20000
-DX_APPLICATION_SECTION_SIZE=2000
-DX_APPLICATION_START=(X_APPLICATION_SECTION_START+X_INTVEC_SIZE)
-DX_APPLICATION_END=(X_APPLICATION_SECTION_START+X_APPLICATION_SECTION_SIZE-1)
// Data (SRAM, external ROM or external NV RAM) memory
-DX_SRAM_BASE=2000 // RAMメモリの開始
-DX_SRAM_TBASE=0 /入 Tiny RAMメモリの開始
-D X_SRAM_TSIZE <mark>#</mark> 0 // Tiny RAMメモリの量
ーDX_SRAM_END=3 <mark>F</mark> FF // <mark>R</mark> AMメモリの最後
//-DX_CSTACK_BASE=X_SRAM_BASE
//-DX_CSTACK_END=X_SRAM_END
//-DX_RSTACK_BASE=X_SRAM_BASE
//-DX_RSTACK_END=X_SPAM_END
// Internal EEPROM
ーDX_EEPROM_END=7PF // EEPROMメモリの最後
-D X_EEPROM_START

Category(区分)でLinker(リンカ)そしてその後に Config(構成設定)タブを選び、link_bootloader_AT xmega128A1.xclを変更した目的対象ファイルに変 更してください。

図1-8. xclファイル選抈	5
Options for node "bo	ootloader" 🛛 🔀
Category: General Options C/C++ Compiler Assembler Custom Build Build Actions Uinker Debugger AVR ONE! CCR ICE200 JTAGICE JTAGICE3 JTAGICE3 JTAGICE3 JTAGICE3 JTAGICE mkII Dragon Simulator Third-Party Driver	Factory Settings Config Output Extra Output List #define Diagnostics Check Image: Check Linker configuration file Image: Check Override Override Override default SpRDJ_DIR\$tink_bootloader_ATxmega128A1.xcl Image: Check Override Default Override program_start Operined Defined Search paths: (one per line) \$TOOLKIT_DIR\$tuBt Image: Check File: Symbol: Segment: Align: Image: Check Symbol:
	OK Cancel

1.4. 手順4 - フラッシュ ページ容量変更

作業空間内でsp_driver.hファイルを選択して FLASH_PAGE_SIZE 512を目的対象デバイ スに対応する正しい値に変更してくださ い。フラッシュ ページ 容量はデータシートで得る ことができます。

図1-9. sp_driver.hでのフラッシュ ページ容量の変更 х main.c sp_driver.h Debug ~ * ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, 0 * (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN A 221 Files * THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILIT. 🖻 🗇 bootloader -... 🗸 defines.h #ifndef SP_DRIVER_H -🕀 💽 eeprom_dri... ÷ #define SP_DRIVER_H – 🗟 eeprom_dri... –🗉 🔂 eeprom_dri... #include "avr_compiler.h" -⊞ 🖸 main.c -⊞ 🖸 serial.c * #include "defines.h" – 🖻 serial.h /* Define the size of the flash page if not defined **₲**sp_driver.h #ifndef FLASH PAGE SIZE #define FLASH_PAGE_SIZE 512 -⊞ 🚮 sp_driver.s90 * 🖵 🖸 Output #endif /*FLASH_PAGE_SIZE*/

作業空間内でsp_driver.s9077イルを選択し てFLASH_PAGE_SIZE 512を目的対象デ バイスに依存する正しい値に変更してくださ い。

図1-10. sp_driver.s90でのフラッシュ ページ容量の変更

Workspace	main.c sp_driver.h sp_driver.s90		
Debug		~	;* LOSS OF USE, DATA, OR PROFITS; OR BUSI
Files	\$	r.	<pre>;* ON ANY THEORY OF LIABILITY, WHETHER IN ** (INCLUDING NEGLIGENCE OF OTHERWISE) AP</pre>
🗉 🗇 bootloader - Debug	¥ .		A THIC COPURATE FIRM IN ADVICED OF THE
🛏 🗟 defines.h			;* INIS SUFIWARE, EVEN IF ADVISED OF INE
⊨⊞ 🖸 eeprom_driver.c			/
🛛 🛏 🖍 eeprom_driver.h			#include <iosyr b=""></iosyr>
🗕 🕀 💼 eeprom_driver_workaround.s90			#include "defines.h"
🛛 🛏 🖸 main.c			
⊨⊞ 🗈 serial.c			/* Define the size of the flash page if n
🛛 🛏 🖻 serial.h			<pre>#ifndef FLASH_PAGE_SIZE</pre>
🛛 🛏 🖻 sp_driver.h			//#error FLASH_PAGE_SIZE_must_be
- ⊕ katisp_driver.s90 >		. * .	#define FLASH_PAGE_SIZE (512)
Len Cutput			<pre>#endif /*FLASH_PAGE_SIZE*/</pre>

1.5. 手順5 - 定義情報変更

作業空間内でdefines.h7ァイルを選択してく ださい。_ATxmega128A1_を目的対象デ ハイスに変更し、その後に予め定義されて いる全ての項目を目的対象デハイスと一致 するように変更してください。満たされた全 ての情報はデータシートで得ることができま す。

図1-11.予め定義されている情報の変更

-		_	main.c sp_driver.h sp_driver.s90 defines.h				
Debug 🗸 🗸							
Files	2: D		#if defined ATxmegal28A1_)				
E 🗇 bootloader - Deb	~		/* definitions for SPM control */				
L Defines h			# define SPMCR_REG NVM.CTRLB				
			# define PAGESIZE 512 //Bytes				
			# define APP_END 0x20000 //Appl				
			/* BLOCKSIZE should be chosen so that the following holds:				
			<pre># define BLOCKSIZE PAGESIZE</pre>				
AlxmegalbD			/* EEPKUM definitions */				
avr_compiler.n			# define EEPRON_NU_PAGES 64				
			# define EEPROM_BYTES_IN_PAGE 32				
DLib_Defaults.h			# define FEDDOM BVTE ADDDESS MASK Ovif				
DLib_Product.h			# define LEPRON_BYTE_ADDRESS_NASK UXIE				
DLib_Threads.h			14 Augustine and Andre accuration 47				
eeprom_driver.h			/* definitions for device recognition */				
📙 ⊨ 🖻 inavr.h			# define PARICUDE UXFA				
│			# define SIGNATURE_DITE_1 UXIE				
📗 🛏 🖻 ioavr.h			# define SIGNATURE_DITE_2 00.97				
📔 📙 🕞 iomacro.h			# deline SIGNATORE_BILE_3 0X40				

1.6. 手順6 - プロジェクトのコンパイル

全てが正しく処理されたなら、プロシェクトは成功裏にコンハイルできます。Project (プロシェクト)⇒Rebuild All(全て再構築)をクリックしてください。すると¥Debug¥Exe フォルダ内に新しいデバック、ファイルの"bootldr.dbg"が作成されるでしょう。このデ バック、ファイルはAtmel Studioでブートローダをデバック、するのに使われます。

図1-12. プロジェクトの再構築

🔀 bootloader - IAR Embedded Workbench IDE						
File Edit View	Project Tools	Window	Help			
Workspace Debug Files Diffues Diffues	Add Files Add Group Import File Li Edit Configur Remove	st ations				
	Create New Add Existing	Project Project				
	Options		Alt+F7			
📙 🖃 💽 main.	Version Cont	rol System	+			
Free C serial	Make Compile Rebuild All	>	F7 Ctrl+F7			
L-⊞ 🗋 Outpu	Clean Batch build		F8			

ブートローダがIAR Embedded Workbenchでデバッグ される場合、出力形式は図1-13.で示されるよう に変更されるべきです。設定後、Project(プロジェ クト)⇒Rebuild All(全て再構築)をクリックしてくださ い。bootloader.d90が生成されるでしょう。このデ パッグ ファイルはブートローダをIAR Embedded Workb enchでデバッグするのに使われます。

図1-13. 出力形式



1.7. 手順7 - ブートロータのティック

フートロータ はIAR Embedded Workbench でデ ハッグすることができます。図1-14.で示される ようにSetup(準備設定)タフ で使うデ ハッカ を選 んでください。



図1-15.で示されるようにProject(プロジェクト)⇒Download and Debug(書き込んで デヾ゙ッヮ´)をクリックしてください。そしてこれは目的対象デヾ゙イスに対するフ゛ートロータ゛ をデヾ゙ッヮ゛します。タ゛ウンロート(書き込み)後、フ゛ートロータ゛のデヾ゙ッヮ゛を開始することが できます。

図1-15. ダウンロート(書き込み)とデバッグ

🔏 bootloader - IAR Embedded Workbench IDE							
File Edit View	Project	JTAGICE3	Tools	Window	Help		
Vorkspace Debug Files D Dottoe	Add F Add C Impol Add F Edit C	Files Group rt File List Project Conn Configuration	ection s				
⊢	Creat Add E Optio	Create New Project Add Existing Project Options Alt+F7					
⊞ [] main. ⊞ [] serial	Version Control System						
┝── ┣ॏ serial ┝── ┣ॏ sp_dr ┝─⊞ ि∰ sp_dr └─⊞ C☐ Outpu	Make Comp Rebu Clear	ile ild All		F7 Ctrl+F	7		
	Batch	ı build		F8			
	Stop	Build		Ctrl+B	reak		
<	Dowr Debu	iload and Del g without Do	bug wnloadi	Ctrl+D ng			

代わりに、bootldr.dbgファイルはAtmel Studio 6でデベッグされるブートロータを 作ることができます。Atmel Studio 6を 開いた後、File(ファイル)⇒Open(開く)⇒ Open Object File For Debugging(デ ベッグ用にオブジェクト ファイルを開く)をク リックし、そしてbootldr.dbgファイルを開く ために¥Debug¥Exeフォルタ゛を閲覧して ください。オブジェクト ファイルでデベッグす る方法のより多くの情報については Atmel Studioヘルプ ファイルを参照してく ださい。

図1-	図1-16. オブジェクト ファイルを開く								
🏶 Start Page - AtmelStudio									
File	Edit View VAssistX ASF Pro	ject Debug	То	ols Window Help					
	New		۲	- 📮 - 🛤 🔚 🍳 🗈 🕅 👘	- 🌁 rc				
	Open		۲	n Project/Solution	Ctrl+Shift+O				
	Close			🚰 File	Ctrl+O				
Ē	Close Solution			Open Object File For Debugging	>				
	Import		۲						
	Save Selected Items	Ctrl+S							
	Save Selected Items As								
1	Save All	Ctrl+Shift+S			1.1.1				
	Export Template								

1.8. 手順8 - bat(バッチ)ファイルの変更

¥AVROSP_Testフォルダを閲覧し、ノートパッド(Notepad)でbatファイルの変更 ルを開いてください。ATxmega128A1を目的対象デバイスに変更 してください。写像ファイル(.hex)はbatファイルと同じフォルダに置か れるべきです。他のbatファイルはそれらが使われるならば同じ方 法で変更されるべきです。

📕 x128A1_flash_write_file.bat - Notepad			
File Edit	Format View Help		
mode co AVROSP pause	m1 Data=8 Parity=n Baud=9600 DTR=OFF RTS=OFF -deTxmega128A1e -if€1ash.he≫ -pf -vf		

2. 推奨読み物

自己プログラミングとブートローダについての全体的な知識を得るために以下の応用記述を読むことが推奨されます。

- AVR109: 自己プログラミング この応用記述はSPM命令を持つデバイスが自己プログラミングに対してどう構成設定され得るかを説明します。これがAtmel tinyAVR[®]とAtmel megaAVR[®]のデバイスに対して与えられるとは言え、それは自己プログラミングについての全般的な情報を与えます。
- AVR1316: XMEGA自己プログラミング この応用記述はAtmel AVR XMEGA自己プログラミングの基本的な機能を記述します。
- AVR1622: XMEGA用TWI7⁻トロー9^{*} この応用記述は応用領域を更新するためにXMEGA系統のデバイスのフ^{*}ートロー9^{*}の使用法と自 己プログラミングに対してXMEGAがどう構成設定され得るかを記述します。
- AVR1605: XMEGAフートローダ即時開始の手引き この応用記述はXMEGA系統デバイスの1つ(換言するとATxmega128A1)でブート ローダ応用を使う方法と自己プログラムに対してプログラムメモリ格納(SPM:Store Prog ram Memory)命令を持つAVR[®]がどう構成設定され得るかを記述します。

3. 改訂履歴

文書改訂	日付	注釈
42153A	2013年7月	初版



Enabling Unlimited Possibilities®

Atmel Corporation

1600 Technology Drive San Jose, CA 95110 USA TEL (+1)(408) 441-0311 FAX (+1)(408) 487-2600 www.atmel.com

Atmel Asia Limited

Unit 01–5 & 16, 19F BEA Tower, Millennium City 5 418 Kwun Tong Road Kwun Tong, Kowloon HONG KONG TEL (+852) 2245–6100 FAX (+852) 2722–1369

Atmel Munich GmbH

Business Campus Parking 4 D-85748 Garching b. Munich GERMANY TEL (+49) 89-31970-0 FAX (+49) 89-3194621

Atmel Japan G.K.

141-0032 東京都品川区 大崎1-6-4 新大崎勧業ビル 16F アトメル ジャパン合同会社 TEL (+81)(3)-6417-0300 FAX (+81)(3)-6417-0370

© 2013 Atmel Corporation. 不許複製 / 改訂:42153A-AVR-07/2013

Atmel[®]、Atmel^{ロコ}とそれらの組み合わせ、Enabling Unlimited Possibilities[®]、AVR[®]、XMEGA[®]とその他はAtmel Corporationの登録 商標または商標またはその付属物です。他の用語と製品名は一般的に他の商標です。

お断り:本資料内の情報はAtmel製品と関連して提供されています。本資料またはAtmel製品の販売と関連して承諾される何れの知 的所有権も禁反言あるいはその逆によって明示的または暗示的に承諾されるものではありません。Atmelのウェブサイトに位置する販売 の条件とAtmelの定義での詳しい説明を除いて、商品性、特定目的に関する適合性、または適法性の暗黙保証に制限せず、Atmel はそれらを含むその製品に関連する暗示的、明示的または法令による如何なる保証も否認し、何ら責任がないと認識します。たと えAtmelがそのような損害賠償の可能性を進言されたとしても、本資料を使用できない、または使用以外で発生する(情報の損失、 事業中断、または利益と損失に関する制限なしの損害賠償を含み)直接、間接、必然、偶然、特別、または付随して起こる如何なる 損害賠償に対しても決してAtmelに責任がないでしょう。Atmelは本資料の内容の正確さまたは完全性に関して断言または保証を行 わず、予告なしでいつでも製品内容と仕様の変更を行う権利を保留します。Atmelはここに含まれた情報を更新することに対してどん な公約も行いません。特に別の方法で提供されなければ、Atmel製品は車載応用に対して適当ではなく、使用されるべきではありま せん。Atmel製品は延命または生命維持を意図した応用での部品としての使用に対して意図、認定、または保証されません。

© *HERO* 2021.

本応用記述はAtmelのAT03160応用記述(Rev.42153A-07/2013)の翻訳日本語版です。日本語では不自然となる重複する形容表現は省略されている場合があります。日本語では難解となる表現は大幅に意訳されている部分もあります。必要に応じて一部加筆されています。頁割の変更により、原本より頁数が少なくなっています。

必要と思われる部分には()内に英語表記や略称などを残す形で表記しています。

青字の部分はリンクとなっています。一般的に赤字の0,1は論理0,1を表します。その他の赤字は重要な部分を表します。