AVR076 : AVR® CAN – 4Kﾌﾞｰﾄﾛｰﾀﾞ

1. 要点

- CAN規約
  - 物理層として使われる制御器域網(CAN)
  - 再設定可能な7つのISP CAN識別子
  - 自動ﾋﾞｯﾄ速度
- 実装書き込み
  - ﾌﾗｯｼｭ ﾒﾓﾘとEEPROMの読み書き
  - ﾃﾞﾊﾞｲｽID読み込み
  - 完全なﾁｯﾌﾟ消去
  - ISP命令からの安全な設定
  - 遠隔応用開始命令
- 応用書き込み
  - 255までの節点(ﾉｰﾄﾞ)
  - 再配置可能な16個の予約識別子
- 応用書き込みｲﾝﾀｰﾌｪｰｽ
  - ﾌﾗｯｼｭ書き込みAPI(応用領域)
- 他の規約を開放
  - LIN
  - RS232
  - SPI
  - TWI
  - その他
- FLIPｲﾝﾀｰﾌｪｰｽ

2. 説明

この資料は"減量"CANﾌﾞｰﾄ ﾛｰﾀﾞの機能だけでなく、ﾁｯﾌﾟ上のﾌﾗｯｼｭ ﾒﾓﾘとEEPROMでの操作を効率的に実行するための規約も記述します。

このﾌﾞｰﾄ ﾛｰﾀﾞは"実装書き込み(ISP:In-System Proguramming)"を実装します。ISPはｼｽﾃﾑからﾃﾞﾊﾞｲｽを取り去ることなく、そして予め書き込まれた応用の必要なく、ﾏｲｸﾛ ｺﾝﾄﾛｰﾗのﾁｯﾌﾟ上のﾌﾗｯｼｭ ﾒﾓﾘとEEPROMの書き込みや書き換えを使用者に許します。

CANﾌﾞｰﾄ ﾛｰﾀﾞはCAN網を通してﾎｽﾄとの通信を管理することができます。それはﾁｯﾌﾟ上のﾌﾗｯｼｭ ﾒﾓﾘとEEPROMでのｱｸｾｽや要求した操作を実行することもできます。

応用書き込み機能は255までのCAN節点(ﾉｰﾄﾞ)の管理が利用可能です。

特殊な入口(ﾌﾗｯｼｭAPI)が使用者に対して利用可能です。

8ﾋﾞｯﾄ AVR®
マイクロ コントローラ

応用記述

AT90CAN32
AT90CAN64
AT90CAN128

ATmega16M1
ATmega32M1
ATmega32C1
ATmega64C1

4Kﾊﾞｲﾄ(減量)
CANﾌﾞｰﾄ ﾛｰﾀﾞ

AVR Studio 4

IAR SYSTEMS

本書は一般の方々の便宜のため有志により作成されたもので、Atmel社とは無関係であることを御承知ください。しおりのはじめにでの内容にご注意ください。

Rev. 8247A-08/09, 8247AJ2-01/21

3. ブート ローダ環境

CANブート ローダはチップ上のフラッシュ メモリの"ブート ローダ フラッシュ領域"に格納されます。ブート ローダの大きさは4Kバイト未満で、故に物理的な"ブート ローダ フラッシュ領域"は全体の半分だけです(訳注:4Kブート領域のデバイスは全部)。応用プログラムの大きさは"応用フラッシュ領域"+4Kバイト(訳注:8Kブート領域のデバイスのみ)と等しいかまたはそれ以下でなければなりません(表3-1.と表3-2.参照)。

表3-1. AT90CANxxx系列-メモリ割付 (バイト アドレス指定)

メモリ種別		AT90CAN128	AT90CAN64	AT90CAN32
フラッシュ メモリ	容量	128Kバイト	64Kバイト	32Kバイト
	アドレス範囲	$00000～$1FFFF	$00000～$0FFFF	$00000～$07FFF
"応用フラッシュ領域"	容量	120Kバイト	56Kバイト	24Kバイト
	アドレス範囲	$00000～$1DFFF	$00000～$0DFFF	$00000～$05FFF
"ブート ローダ フラッシュ領域"	容量	8Kバイト		
	アドレス範囲	$1E000～$1FFFF	$0E000～$0FFFF	$06000～$07FFF
"ブート ローダ リセット アドレス" (注1)	最小(第1)ブート	$1FC00	$0FC00	$07C00
	第2ブート	$1F800	$0F800	$07800
	第3ブート	$1F000 (注2)	$0F000 (注2)	$07000 (注2)
	最大(第4)ブート	$1E000	$0E000	$06000
EEPROM	容量	4Kバイト	2Kバイト	1Kバイト
	アドレス範囲	$0000～$0FFF	$0000～$07FF	$0000～$03FF

注1: "ブート ローダ リセット アドレス"は"BOOTSZ"ヒューズ ビットに依存します。メモリ(フラッシュ メモリ、EEPROMなど)の動きのより多くの詳細についてはデータシートを参照してください。

注2: CANブート ローダ リセット アドレス

表3-2. ATmegaxxM1/C1系列-メモリ割付 (バイト アドレス指定)

メモリ種別		ATmega64M1/C1	ATmega32M1/C1	ATmega16M1
フラッシュ メモリ	容量	64Kバイト	32Kバイト	16Kバイト
	アドレス範囲	$0000～$FFFF	$0000～$7FFF	$0000～$3FFF
"応用フラッシュ領域"	容量	$0000～$7FFF	28Kバイト	12Kバイト
	アドレス範囲	$0000～$DFFF	$0000～$6FFF	$0000～$2FFF
"ブート ローダ フラッシュ領域"	容量	8Kバイト	4Kバイト	
	アドレス範囲	$E000～$FFFF	$7000～$7FFF	$3000～$3FFF
"ブート ローダ リセット アドレス" (注1)	最小(第1)ブート	$FC00	$7E00	$3E00
	第2ブート	$F800	$7C00	$3C00
	第3ブート	$F000 (注2)	$7800	$3800
	最大(第4)ブート	$E000	$7000 (注2)	$3000 (注2)
EEPROM	容量	2Kバイト	1Kバイト	512バイト
	アドレス範囲	$0000～$07FF	$0000～$03FF	$0000～$01FF

注1: "ブート ローダ リセット アドレス"は"BOOTSZ"ヒューズ ビットに依存します。メモリ(フラッシュ メモリ、EEPROMなど)の動きのより多くの詳細についてはデータシートを参照してください。

注2: CANブート ローダ リセット アドレス

3.1. ﾃﾞﾊﾞｲｽ ﾋｭｰｽﾞ設定

更なる説明についてはﾃﾞﾊﾞｲｽのﾃﾞｰﾀｼｰﾄを参照してください。

図3-1. ﾃﾞﾊﾞｲｽ ﾋｭｰｽﾞ設定 – 第1部

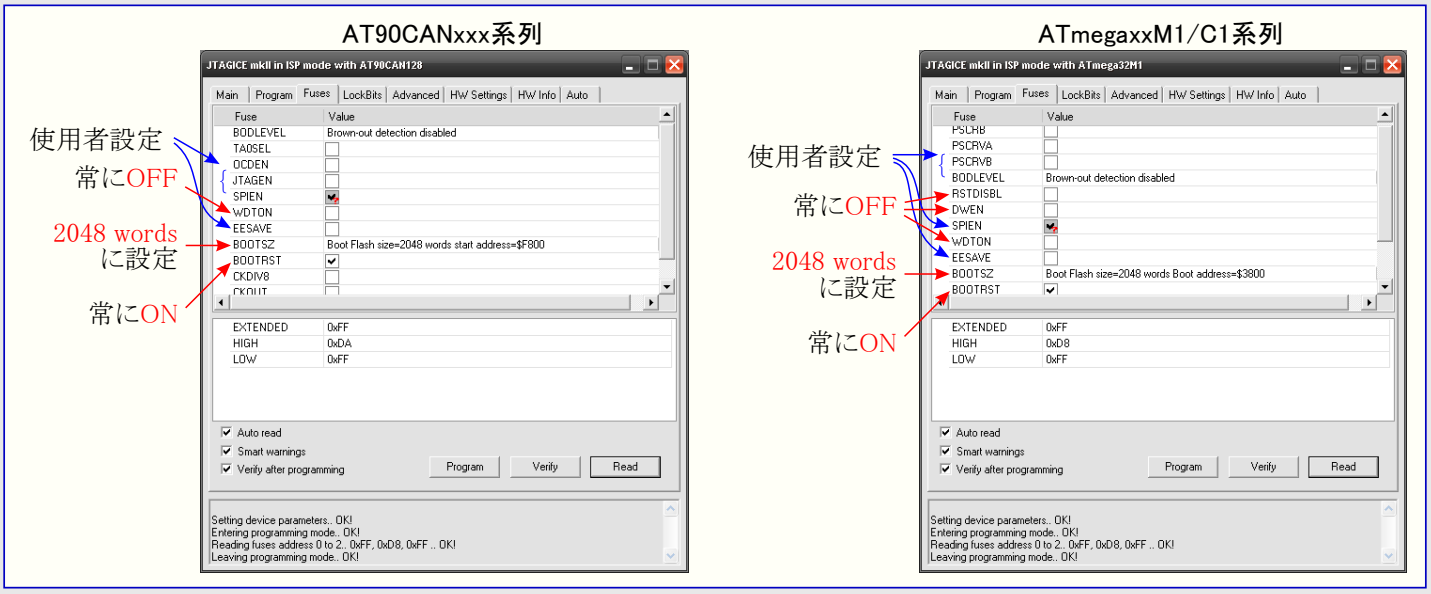
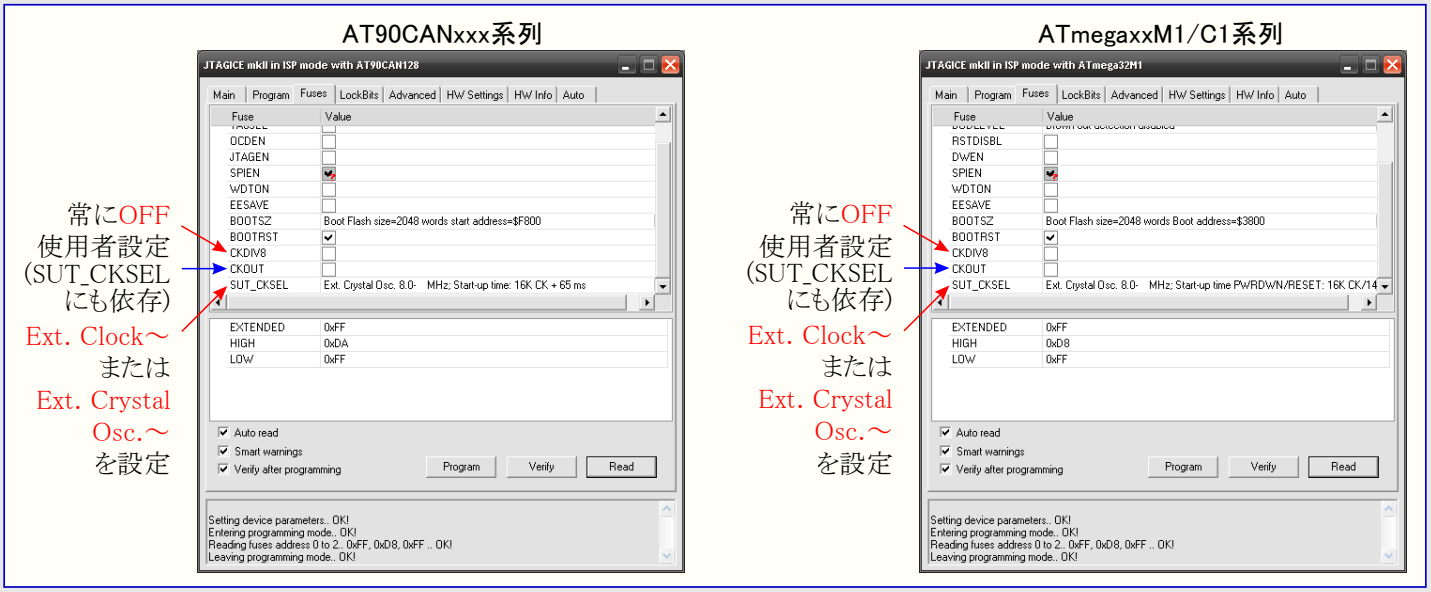


図3-2. ﾃﾞﾊﾞｲｽ ﾋｭｰｽﾞ設定 – 第2部



必須ﾋｭｰｽﾞ設定の要約:

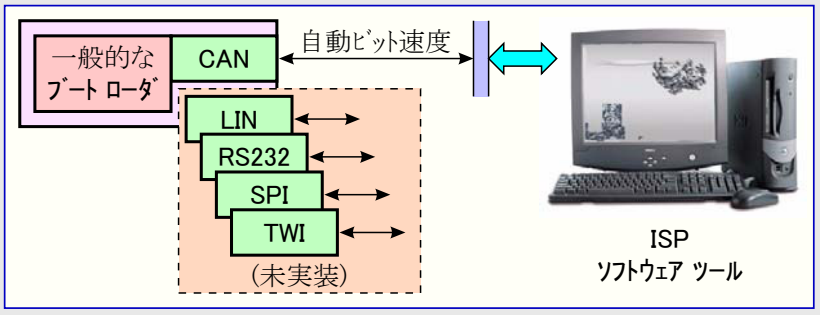
ﾋｭｰｽﾞ上位ﾊﾞｲﾄ： BOOTRST ﾌﾟﾛｸﾞﾗﾑ(0)
BOOTSZ1,0 2048語に設定
WDTON 非ﾌﾟﾛｸﾞﾗﾑ(1)

ﾋｭｰｽﾞ下位ﾊﾞｲﾄ： CKSEL3～0 CAN必要条件に合う高い精度を持つｸﾛｯｸを選択するように設定(内蔵RC発振器は合致しません。)

3.2. 物理的な環境

一般的なﾌﾞｰﾄ ﾛｰﾀﾞはCANｲﾝﾀｰﾌｪｰｽを通してﾎｽﾄ(またはPC)を処理します。一般的なﾌﾞｰﾄ ﾛｰﾀﾞは他のｲﾝﾀｰﾌｪｰｽ(LIN,RS232, SPI,TWI,～)に接続することができるｻｰﾋﾞｽです。

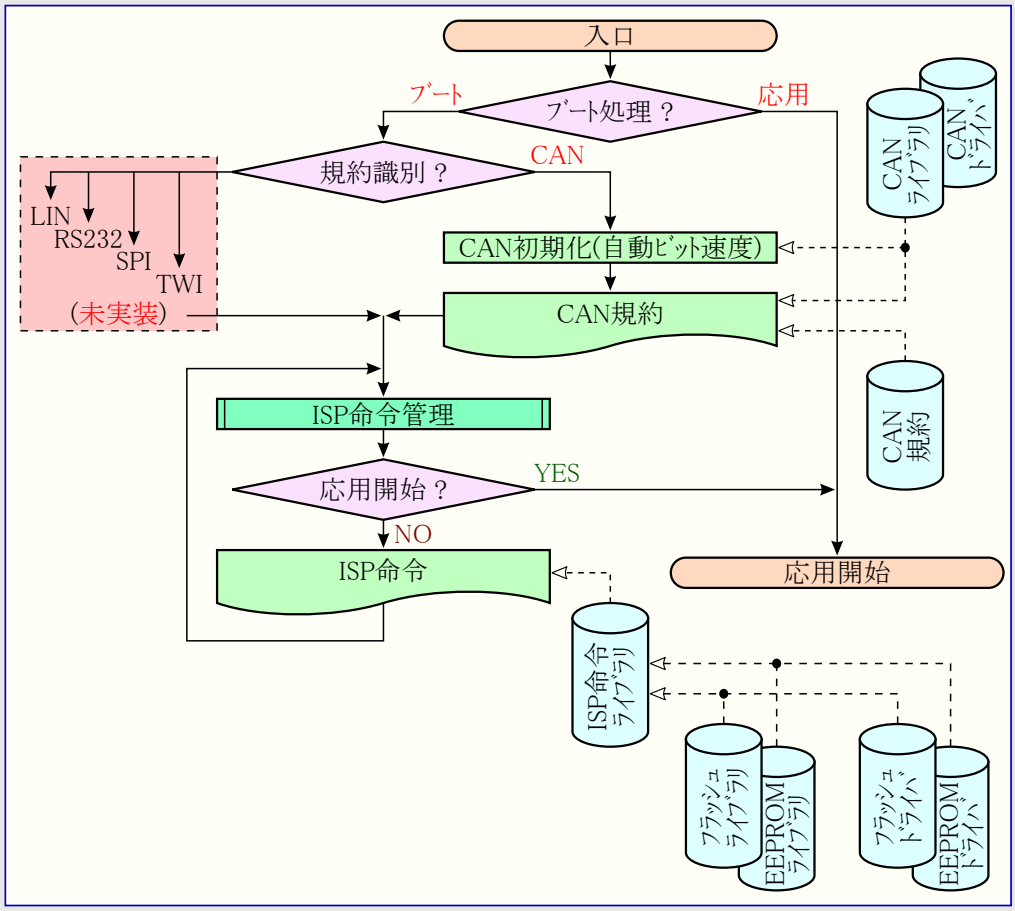
図3-3. 物理的な環境



3.3. ﾌﾞｰﾄ ﾛｰﾀﾞ説明

3.3.1. 概要

図3-4. ﾌﾞｰﾄ ﾛｰﾀﾞ構成図



3.3.2. 入口点

1つの"入口点"だけが利用可能で、それはﾌﾞｰﾄ ﾛｰﾀﾞへの入口点です。ﾃﾞﾊﾞｲｽの"BOOTRST"ﾋｭｰｽﾞが設定(0)されなければなりません。ﾘｾｯﾄ後、ﾃﾞﾊﾞｲｽの"ﾌﾟﾛｸﾞﾗﾑ ｶｳﾝﾀ"は"ﾌﾞｰﾄ ﾘｾｯﾄ ｱﾄﾞﾚｽ"に設定されます(2頁の表3-1. AT90CANxxx系列-ﾒﾓﾘ割付(ﾊﾞｲﾄ ｱﾄﾞﾚｽ指定)と表3-2. ATmegaxxM1/C1系列-ﾒﾓﾘ割付(ﾊﾞｲﾄ ｱﾄﾞﾚｽ指定)参照)。この"入口点"はﾌﾞｰﾄ ﾛｰﾀﾞの"ﾌﾞｰﾄ処理"を初期化します。

3.3.3. ﾌﾞｰﾄ処理

ﾌﾞｰﾄ ﾛｰﾀﾞの"ﾌﾞｰﾄ処理"は応用またはﾌﾞｰﾄ ﾛｰﾀﾞ自身の開始を許します。これは次の2つの変数に依存します。

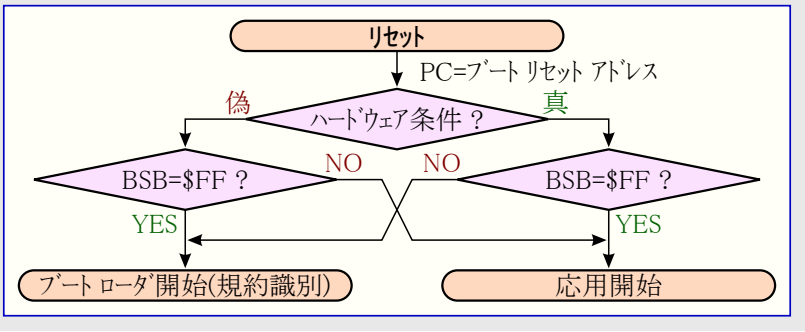
- **"ﾊｰﾄﾞｳｪｱ条件"**

ﾊｰﾄﾞｳｪｱ条件は(このﾌﾞｰﾄ ﾛｰﾀﾞでHWCBと名付けられた)ﾃﾞﾊﾞｲｽの入力ﾋﾟﾝとそれの活性ﾚﾍﾞﾙ(例:INT0/PIND.0, active low)によって定義されます。

- **"ﾌﾞｰﾄ状態ﾊﾞｲﾄ"**

ﾌﾞｰﾄ状態ﾊﾞｲﾄ"BSB(Boot Status Byte)"は"ﾌﾞｰﾄ ﾛｰﾀﾞ構成設定ﾒﾓﾘ"に属します(8頁の「4.5.4.1. ﾌﾞｰﾄ状態ﾊﾞｲﾄ – "BSB"」項参照)。これの既定値は$FFです。ISP命令はこの値の変更を許します。

図3-5. ﾌﾞｰﾄ処理構成図



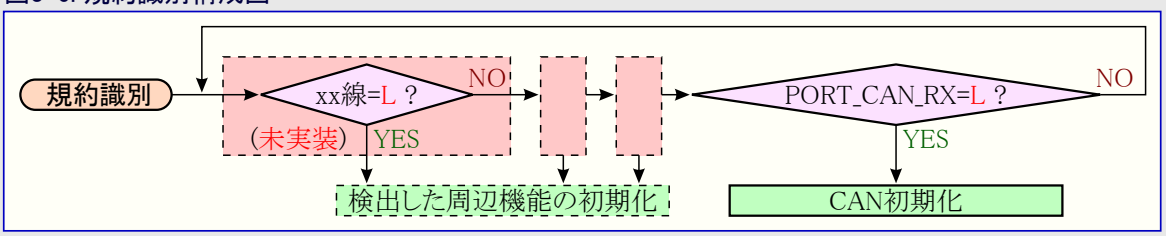
3.3.4. 規約識別

ﾌﾞｰﾄ ﾛｰﾀﾞの"規約識別"は使われる何かの規約、CANまたは他の規約を選びます。媒体での活動を検知するために物理(信号)線のﾎﾟｰﾘﾝｸﾞが行われます。それらの線は次のとおりです。

- PORT_CAN_RX：以下でﾎﾟｰﾘﾝｸﾞが行われます。
  - AT90CANxxx系列についてはRXCAN/PIND.6
  - ATmegaxxM1/C1系列についてはRXCAN/PINC.3
- (使用者定義ｲﾝﾀｰﾌｪｰｽ)

PORT_CAN_RX線のLowﾚﾍﾞﾙがCAN周辺機能の初期化を開始します。

図3-6. 規約識別構成図



3.3.5. CAN初期化

ﾎｽﾄとの通信に使われるCANは以下の構成設定を持ちます。

- **規格**　　：CAN形式2.0A (11ﾋﾞｯﾄ識別子)
- **ﾌﾚｰﾑ**　　：ﾃﾞｰﾀ ﾌﾚｰﾑ
- **ﾋﾞｯﾄ速度**：特別ﾊﾞｲﾄ(EB:Extra Byte)に依存(9頁の「特別ﾊﾞｲﾄ – "EB"」をご覧ください。)
  - "EB"＝$FF：ｿﾌﾄｳｪｱ自動ﾋﾞｯﾄ速度を使用
  - "EB"≠$FF：CANﾋﾞｯﾄ速度設定にﾋﾞｯﾄ ﾀｲﾐﾝｸﾞ制御1～3を使用(9頁の「ﾋﾞｯﾄ ﾀｲﾐﾝｸﾞ制御1～3 – "BTC1～3"」をご覧ください。)

初期化処理は各ﾃﾞﾊﾞｲｽ ﾘｾｯﾄ後に実行されなければなりません。ﾎｽﾄは節点(ﾉｰﾄﾞ)を選択するためにﾃﾞｰﾀ ﾌﾚｰﾑを送ることによって通信を始めます。自動ﾋﾞｯﾄ速度の場合、これがCANﾋﾞｯﾄ速度を見つけるため、ﾌﾞｰﾄ ﾛｰﾀﾞを助けます。CAN規格は異常応答を持つﾌﾚｰﾑは自動的に再送されると言っています。この機能と"聴取"形態に設定されるべきCAN周辺機能の能力が自動ﾋﾞｯﾄ速度によって使われます。一旦同期ﾌﾚｰﾑがどの異常もなく受信されると、"聴取"形態の開放によって応答で劣性ﾚﾍﾞﾙが印加されます。

ｿﾌﾄｳｪｱ自動ﾋﾞｯﾄ速度はﾃﾞﾊﾞｲｽで設定されたｼｽﾃﾑ ｸﾛｯｸ(CKIO)に従って広範囲のﾎﾞｰﾚｰﾄを支援します("config.h"ﾌｧｲﾙで"FOC"定義参照)。この(自動ﾋﾞｯﾄ速度)機能は多数のCAN節点(ﾉｰﾄﾞ)を持つCAN網で保証されません。この場合に於いては固定ﾎﾞｰﾚｰﾄ("EB"≠$FF)が推奨されます。

3.3.6. CAN規約概要

"CAN規約"は直列線(CANﾊﾞｽ)上の上位規約です。

これは本資料の特別な節で記述されます(11頁の「**CAN規約とISP命令**」をご覧ください)。

3.3.7. ISP命令概要

"CAN規約"は"ISP命令"を復号します。"ISP命令"の組はどの規約とも明らかに無関係です。

これは本資料の特別な節で記述されます(11頁の「**CAN規約とISP命令**」をご覧ください)。

3.3.8. ﾌﾞｰﾄ ﾛｰﾀﾞからの出力

ﾌﾞｰﾄ ﾛｰﾀﾞからの出力はISP命令:"応用開始"を受信した後で実行されます。(11頁の「**CAN規約とISP命令**」をご覧ください)。

4. メモリ空間定義

ﾌﾞｰﾄ ﾛｰﾀﾞは6つまでの独立したﾒﾓﾘ空間を支援します。それらの各々は低位ｱｸｾｽ規約(ﾄﾞﾗｲﾊﾞ)が異なり得るため、ｺｰﾄﾞ番号(対応する規約領域で報告される値)を受け取ります。

ﾒﾓﾘ空間のｱｸｾｽはﾊﾞｲﾄ ｱｸｾｽ(換言すると、ﾊﾞｲﾄ ｱﾄﾞﾚｽで与えられた

表4-1. ﾒﾓﾘ空間ｺｰﾄﾞ番号

空間　(注)	ｺｰﾄﾞ番号	ｱｸｾｽ
ﾌﾗｯｼｭ ﾒﾓﾘ	0	読み書き
EEPROMﾃﾞｰﾀ ﾒﾓﾘ	1	読み書き
識票	2	読み込み専用
ﾌﾞｰﾄ ﾛｰﾀﾞ情報	3	読み込み専用
ﾌﾞｰﾄ ﾛｰﾀﾞ構成設定	4	読み書き
ﾃﾞﾊﾞｲｽ ﾚｼﾞｽﾀ	5	読み込み専用

注: 時々、識別は物理的ではありません(例:"識票"は"ﾌﾞｰﾄ ﾛｰﾀﾞ情報"だけでなく"ﾌﾞｰﾄ ﾛｰﾀﾞ ﾌﾗｯｼｭ領域"のｺｰﾄﾞの部分でもあります)。

4.1. ﾌﾗｯｼｭ ﾒﾓﾘ空間

ﾌﾞｰﾄ ﾛｰﾀﾞによって管理されるﾌﾗｯｼｭ ﾒﾓﾘ空間はﾃﾞﾊﾞｲｽのﾌﾗｯｼｭ ﾒﾓﾘの部分です。それは"応用ﾌﾗｯｼｭ領域"です。

表4-2. ﾌﾗｯｼｭ ﾒﾓﾘ空間 (ｺｰﾄﾞ番号0)

ﾌﾗｯｼｭ ﾒﾓﾘ空間	AT90CAN128	AT90CAN64 ATmega64M1 ATmega64C1	AT90CAN32 ATmega32M1 ATmega32C1	ATmega16M1
容量	124Kﾊﾞｲﾄ	60Kﾊﾞｲﾄ	28Kﾊﾞｲﾄ	12Kﾊﾞｲﾄ
ｱﾄﾞﾚｽ範囲	$00000～$1EFFF	$0000～$EFFF	$0000～$6FFF	$0000～$2FFF
ﾍﾟｰｼﾞ数　(注)	2	1	1	1

注: ﾍﾟｰｼﾞ ﾊﾟﾗﾒｰﾀはﾌﾞｰﾄ ﾛｰﾀﾞとﾃﾞﾊﾞｲｽ自体で異なります。

4.1.1. 読み書き

"ISP読み込み"または"ISP書き込み"命令だけが64Kﾊﾞｲﾄのﾍﾟｰｼﾞ内にﾊﾞｲﾄ ｱﾄﾞﾚｽ指定形態でﾌﾗｯｼｭ ﾒﾓﾘ空間にｱｸｾｽします(**表4-2. ﾌﾗｯｼｭ ﾒﾓﾘ空間** (ｺｰﾄﾞ番号0)参照)。特定のISP命令が異なるﾍﾟｰｼﾞの選択を許します。

読み書き命令が起きる間にｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ"SSB(Software Security Byte)"が設定されている場合、ﾌﾞｰﾄ ﾛｰﾀﾞは"ﾃﾞﾊﾞｲｽ保護"異常を返します(9頁の「**4.5.4.2. ｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ - "SSB"**」項参照)。

4.1.2. 消去

"ISP消去"命令はﾌﾗｯｼｭ ﾒﾓﾘ空間の完全消去(全ﾊﾞｲﾄ=$FF)です。この操作は例えｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ"SSB"が設定されていても利用可能です。この操作の終わりで、ｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ"SSB"は保護ﾚﾍﾞﾙ0にﾘｾｯﾄされます(9頁の「**4.5.4.2. ｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ - "SSB"**」項)。

4.1.3. 制限

ﾌﾗｯｼｭ ﾒﾓﾘ空間でのISP命令はﾌﾞｰﾄ ﾛｰﾀﾞに於いて無効です("ﾌﾞｰﾄ ﾛｰﾀﾞ ﾌﾗｯｼｭ領域"で無効)。

ISP命令に対するﾌﾗｯｼｭ ﾒﾓﾘ空間(ｺｰﾄﾞ番号0)の大きさは「**表4-2. ﾌﾗｯｼｭ ﾒﾓﾘ空間** (ｺｰﾄﾞ番号0)」で与えられます。

4.2. EEPROMﾃﾞｰﾀ ﾒﾓﾘ

ﾌﾞｰﾄ ﾛｰﾀﾞによって管理されるEEPROMﾃﾞｰﾀ ﾒﾓﾘ空間はﾃﾞﾊﾞｲｽのEEPROMです。

表4-3. EEPROMﾃﾞｰﾀ ﾒﾓﾘ空間 (ｺｰﾄﾞ番号1)

EEPROM ﾃﾞｰﾀ ﾒﾓﾘ空間	AT90CAN128	AT90CAN64 ATmega64M1 ATmega64C1	AT90CAN32 ATmega32M1 ATmega32C1	ATmega16M1
容量	4Kﾊﾞｲﾄ	2Kﾊﾞｲﾄ	1Kﾊﾞｲﾄ	512ﾊﾞｲﾄ
ｱﾄﾞﾚｽ範囲	$0000～$0FFF	$0000～$07FF	$0000～$03FF	$0000～$01FF
ﾍﾟｰｼﾞ数	(ﾍﾟｰｼﾞなし)			

4.2.1. 読み書き

EEPROMﾃﾞｰﾀ ﾒﾓﾘ空間は不揮発性ﾃﾞｰﾀ ﾒﾓﾘとして使われます。"ISP読み込み"または"ISP書き込み"命令はこの空間に(ﾍﾟｰｼﾞなしで)ﾊﾞｲﾄ単位でｱｸｾｽします。

読み書き命令が起きる間にｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ"SSB(Software Security Byte)"が設定されている場合、ﾌﾞｰﾄ ﾛｰﾀﾞは"ﾃﾞﾊﾞｲｽ保護"異常を返します(9頁の「4.5.4.2. ｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ – "SSB"」項参照)。

4.2.2. 消去

"ISP消去"命令はEEPROMﾃﾞｰﾀ ﾒﾓﾘ空間の完全消去(全ﾊﾞｲﾄ=$FF)です。この操作はｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ"SSB"がﾘｾｯﾄされている場合にだけ利用可能です(9頁の「4.5.4.2. ｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ – "SSB"」項)。

4.2.3. 制限

ISP命令に対するEEPROMﾃﾞｰﾀ ﾒﾓﾘ空間(ｺｰﾄﾞ番号1)の大きさは「表4-3. EEPROMﾃﾞｰﾀ ﾒﾓﾘ空間 (ｺｰﾄﾞ番号1)」で与えられます。

4.3. 識票

ﾌﾞｰﾄ ﾛｰﾀﾞによって管理される識票空間はﾌﾞｰﾄ ﾛｰﾀﾞのｺｰﾄﾞに含まれます。これは"ﾌﾞｰﾄ ﾛｰﾀﾞ ﾌﾗｯｼｭ領域"内です。

表4-4. 識票空間 (ｺｰﾄﾞ番号2)

識票空間		AT90 CAN128	AT90 CAN64	AT90 CAN32	ATmega 64M1	ATmega 32M1	ATmega 16M1	ATmega 64C1	ATmega 32C1
製造者符号	ｱﾄﾞﾚｽ:$00 (読み込み専用)	$1E							
系統符号	ｱﾄﾞﾚｽ:$01 (読み込み専用)	$81			$84			$86	
製品名	ｱﾄﾞﾚｽ:$02 (読み込み専用)	$97	$96	$95	$96	$95	$94	$96	$95
製品改訂	ｱﾄﾞﾚｽ:$03 (読み込み専用)	$00							
ﾍﾟｰｼﾞ数		(ﾍﾟｰｼﾞなし)							

4.3.1. 読み書き

"ISP読み込み"命令はこの空間に(ﾍﾟｰｼﾞなしで)ﾊﾞｲﾄ単位でｱｸｾｽします。

ｱｸｾｽ保護はこの読み込み専用空間で全く提供されません。

4.3.2. 消去

読み込み専用空間のために当てはまりません。

4.3.3. 制限

ISP命令に対する識票空間(ｺｰﾄﾞ番号2)の詳細は「表4-4. 識票空間 (ｺｰﾄﾞ番号2)」で与えられます。

4.4. ﾌﾞｰﾄ ﾛｰﾀﾞ情報

ﾌﾞｰﾄ ﾛｰﾀﾞによって管理されるﾌﾞｰﾄ ﾛｰﾀﾞ情報空間はﾌﾞｰﾄ ﾛｰﾀﾞのｺｰﾄﾞに含まれます。これは"ﾌﾞｰﾄ ﾛｰﾀﾞ ﾌﾗｯｼｭ領域"内です。

表4-5. ﾌﾞｰﾄ ﾛｰﾀﾞ情報空間 (ｺｰﾄﾞ番号3)

ﾌﾞｰﾄ ﾛｰﾀﾞ情報空間		AT90 CAN128	AT90 CAN64	AT90 CAN32	ATmega 64M1	ATmega 32M1	ATmega 16M1	ATmega 64C1	ATmega 32C1
ﾌﾞｰﾄ ﾛｰﾀﾞ改訂	ｱﾄﾞﾚｽ:$00 (読み込み専用)	$01							
ﾌﾞｰﾄ識別1	ｱﾄﾞﾚｽ:$01 (読み込み専用)	$D1							
ﾌﾞｰﾄ識別2	ｱﾄﾞﾚｽ:$02 (読み込み専用)	$D2							
ﾍﾟｰｼﾞ数		(ﾍﾟｰｼﾞなし)							

4.4.1. 読み書き

"ISP読み込み"命令はこの空間に(ﾍﾟｰｼﾞなしで)ﾊﾞｲﾄ単位でｱｸｾｽします。
ｱｸｾｽ保護はこの読み込み専用空間で全く提供されません。

4.4.2. 消去

読み込み専用空間のために当てはまりません。

4.4.3. 制限

ISP命令に対するﾌﾞｰﾄ ﾛｰﾀﾞ情報空間(ｺｰﾄﾞ番号3)の詳細は「表4-5. ﾌﾞｰﾄ ﾛｰﾀﾞ情報空間 (ｺｰﾄﾞ番号3)」で与えられます。

4.4.4. ﾌﾞｰﾄ ﾛｰﾀﾞ情報ﾊﾞｲﾄ説明

4.4.4.1. ﾌﾞｰﾄ改訂

ﾌﾞｰﾄ改訂：読み込み専用ｱﾄﾞﾚｽ=$00、値=$01

4.4.4.2. ﾌﾞｰﾄ識別1,2

ﾌﾞｰﾄ識別1,2：読み込み専用ｱﾄﾞﾚｽ=$01,$02、値=$D1,$D2

4.5. ﾌﾞｰﾄ ﾛｰﾀﾞ構成設定

ﾌﾞｰﾄ ﾛｰﾀﾞによって管理されるﾌﾞｰﾄ ﾛｰﾀﾞ構成設定空間は"ﾌﾞｰﾄ ﾛｰﾀﾞ ﾌﾗｯｼｭ領域"に含まれます。

表4-6. ﾌﾞｰﾄ ﾛｰﾀﾞ構成設定空間 (ｺｰﾄﾞ番号4)

ﾌﾞｰﾄ ﾛｰﾀﾞ構成設定空間			既定値
ﾌﾞｰﾄ状態ﾊﾞｲﾄ	"BSB"	ｱﾄﾞﾚｽ:$00	$FF
ｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ	"SSB"	ｱﾄﾞﾚｽ:$01	$FF
特別ﾊﾞｲﾄ	"EB"	ｱﾄﾞﾚｽ:$02	$FF(注1)
ﾋﾞｯﾄ ﾀｲﾐﾝｸﾞ制御1	"BTC1"	ｱﾄﾞﾚｽ:$03	$FF(注2)
ﾋﾞｯﾄ ﾀｲﾐﾝｸﾞ制御2	"BTC2"	ｱﾄﾞﾚｽ:$04	$FF(注2)
ﾋﾞｯﾄ ﾀｲﾐﾝｸﾞ制御3	"BTC3"	ｱﾄﾞﾚｽ:$05	$FF(注2)
節点(ﾉｰﾄﾞ)番号	"NNB"	ｱﾄﾞﾚｽ:$06	$FF(注3)
CAN再配置ID区分	"CRIS"	ｱﾄﾞﾚｽ:$07	$00
開始ｱﾄﾞﾚｽ下位	"SA_L"	ｱﾄﾞﾚｽ:$08	$00
開始ｱﾄﾞﾚｽ上位	"SA_H"	ｱﾄﾞﾚｽ:$09	$00
ﾍﾟｰｼﾞ数			(ﾍﾟｰｼﾞなし)

注1: 効力については9頁の「特別ﾊﾞｲﾄ – "EB"」をご覧ください。
注2: 効力については9頁の「ﾋﾞｯﾄ ﾀｲﾐﾝｸﾞ制御1～3 – "BTC1～3"」をご覧ください。
注3: 効力については9頁の「(CAN)節点(ﾉｰﾄﾞ)番号 – "NNB"」をご覧ください。

4.5.1. 読み書き

"ISP読み込み"命令はこの空間に(ﾍﾟｰｼﾞなしで)ﾊﾞｲﾄ単位でｱｸｾｽします。
ｱｸｾｽ保護はｿﾌﾄｳｪｱ保護ﾊﾞｲﾄでだけ提供されます(9頁の「4.5.4.2. ｿﾌﾄｳｪｱ保護ﾊﾞｲﾄ – "SSB"」項参照)。

4.5.2. 消去

"ISP消去"命令はこの空間に対して利用できません。

4.5.3. 制限

ISP命令に対するﾌﾞｰﾄ ﾛｰﾀﾞ構成設定空間(ｺｰﾄﾞ番号4)の詳細は「表4-5. ﾌﾞｰﾄ ﾛｰﾀﾞ構成設定空間 (ｺｰﾄﾞ番号4)」で与えられます。

4.5.4. ﾌﾞｰﾄ ﾛｰﾀﾞ構成設定ﾊﾞｲﾄ説明

4.5.4.1. ﾌﾞｰﾄ状態ﾊﾞｲﾄ – "BSB"

ﾌﾞｰﾄ ﾛｰﾀﾞのﾌﾞｰﾄ状態ﾊﾞｲﾄは応用またはﾌﾞｰﾄ ﾛｰﾀﾞの開始を制御するために、"ﾌﾞｰﾄ処理"(5頁の「3.3.3. ﾌﾞｰﾄ処理」項)で使われます。ﾊｰﾄﾞｳｪｱ条件が全く設定されない場合、ﾌﾞｰﾄ状態ﾊﾞｲﾄの既定値($FF)はﾌﾞｰﾄ ﾛｰﾀﾞの開始を強制し、さもなければ(ﾌﾞｰﾄ状態ﾊﾞｲﾄ≠$FF且つﾊｰﾄﾞｳｪｱ条件なし)応用が始まります。

4.5.4.2. ソフトウェア保護バイト – "SSB"

ブート ローダは使用者アクセスまたはISPアクセスから自身と応用を保護するためにソフトウェア保護バイト"SSB"を持ちます。これはフラッシュ メモリとEEPROM空間と自身を保護します。

ソフトウェア保護バイト"SSB"での"ISP書き込み"命令はより高い優先レベルだけを書くことができます。3つの保護レベルがあります。

表4-7. 保護レベル

レベル	保護	"SSB"	注釈
0	NO_SECURITY	$FF	• これが既定レベルです。 • レベル1またはレベル2だけがレベル0を上書きすることができます。
1	WR_SECURITY	$FE	• レベル1ではフラッシュ メモリとEEPROMの空間で書くことができません。 • ブート ローダは異常メッセージを返します。 • レベル2だけがレベル1を上書きすることができます。
2	RD_WR_SECURITY	$FC	• フラッシュ メモリとEEPROMの空間での全ての読み書きアクセスが許されません。 • ブート ローダは異常メッセージを返します。 • フラッシュ メモリ空間での"ISP消去"命令だけがソフトウェア保護バイトを(レベル0に)リセットします。

下表はSSBレベルに関して認められた活動を与えます。

表4-8. ソフトウェア保護バイト"SSB"に関して許された活動

ISP命令	NO_SECURITY	WR_SECURITY	RD_WR_SECURITY
フラッシュ メモリ空間消去	許可	許可	許可
EEPROM空間消去	許可	–	–
フラッシュ メモリ空間書き込み	許可	–	–
EEPROM空間書き込み	許可	–	–
フラッシュ メモリ空間読み込み	許可	許可	–
EEPROM空間読み込み	許可	許可	–
("SSB"を除き)ブート ローダ構成設定書き込み	許可	–	–
ブート ローダ構成設定読み込み	許可	許可	許可
"SSB"書き込み	許可	より高いレベルのみ	–
ブート ローダ情報読み込み	許可	許可	許可
識票読み込み	許可	許可	許可
(何れかのメモリの)空白検査	許可	許可	許可
メモリ空間変更	許可	許可	許可

4.5.4.3. 特別バイト – "EB"

特別バイトはCAN初期化を自動ビット速度または固定のCANビット タイミングに切り替えるのに用いられます。

- "EB"＝$FF：ソフトウェア自動ビット速度使用
- "EB"≠$FF：(自動ビット速度ではなく)CAN周辺機能のCANビット タイミング レジスタを設定するのにブート ローダ構成設定空間のCANBT1～3バイトを使用

4.5.4.4. ビット タイミング制御1～3 – "BTC1～3"

"EB"≠$FFの時にCAN周辺機能のCANビット タイミング レジスタを設定するのにブート ローダ構成設定空間のビット タイミング制御1～3バイト("BTC1","BTC2","BTC3")が使われます(自動ビット速度ではありません)。

これらのバイトを構成設定する方法は10頁の「4.6.4.1. CANBT1～3レジスタ」項で記述されます。

4.5.4.5. (CAN)節点(ノード)番号 – "NNB"

11頁の「CAN規約とISP命令」をご覧ください。

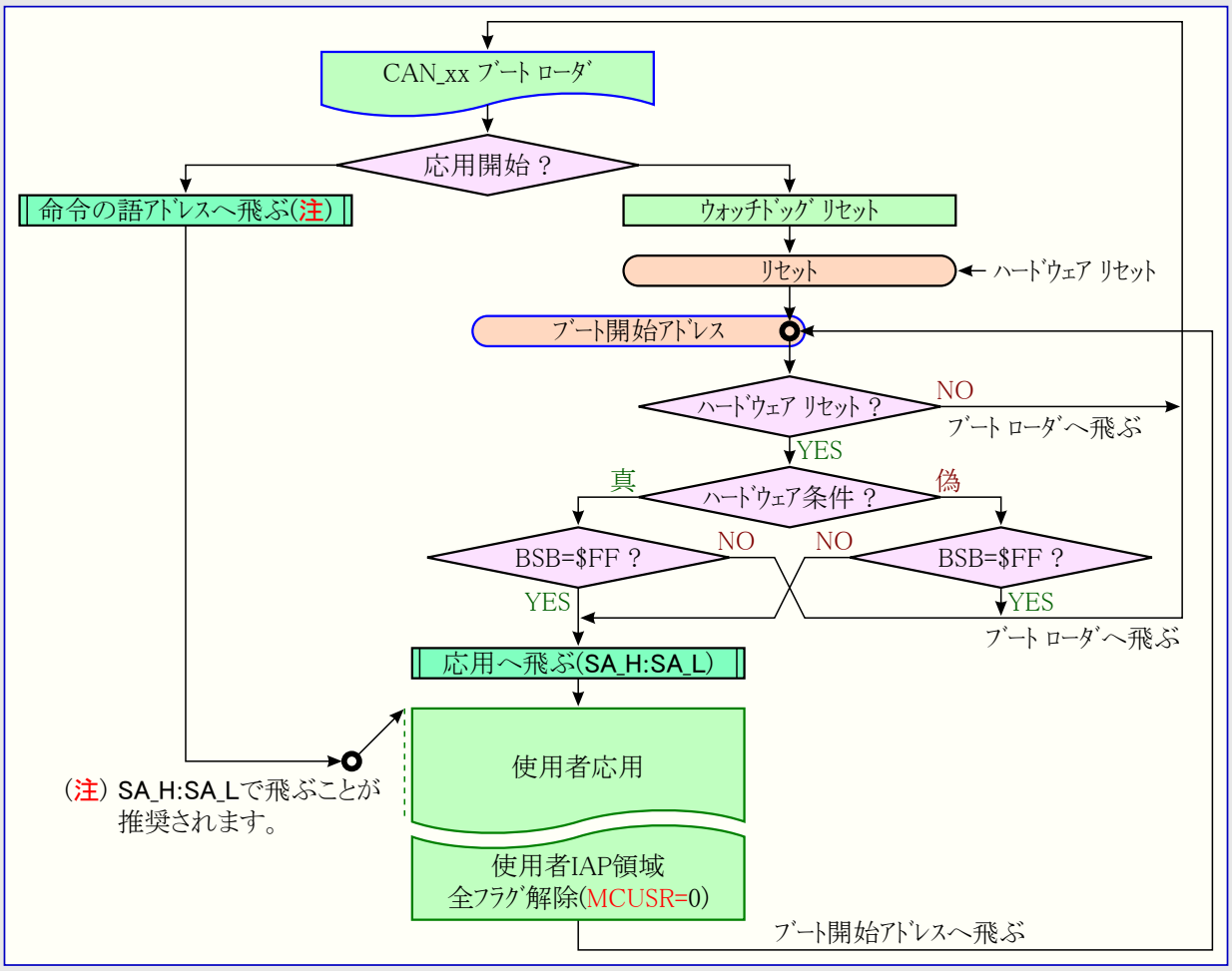
4.5.4.6. CAN再配置ID区分 – "CRIS"

11頁の「CAN規約とISP命令」をご覧ください。

4.5.4.7. 開始(応用)アドレス上位と下位 – "SA_H"と"SA_L"

11頁の「CAN規約とISP命令」をご覧ください。

図4-1. 応用開始とﾘｾｯﾄの構成図



4.6. ﾃﾞﾊﾞｲｽ ﾚｼﾞｽﾀ

ﾌﾞｰﾄ ﾛｰﾀﾞによって管理されるﾃﾞﾊﾞｲｽ ﾚｼﾞｽﾀ空間(ｺｰﾄﾞ番号5)はﾃﾞﾊﾞｲｽの64個の標準I/Oﾚｼﾞｽﾀと160個の拡張I/Oﾚｼﾞｽﾀです。これらは等価なｱｾﾝﾌﾞﾘ言語命令によってｱｸｾｽされます。

```
LDS     Rxx, REG_ADD          ここでREG_ADDは次のｱﾄﾞﾚｽ範囲です。
```

- AT90CANxxx系列については$20(PINA)～$FA(CANMSG)
- ATmegaxxM1/C1系列については$23(PINB)～$FA(CANMSG)

4.6.1. 読み書き

"ISP読み込み"命令はこの空間に(ﾍﾟｰｼﾞなしで)ﾊﾞｲﾄ単位でｱｸｾｽします。

ｱｸｾｽ保護はこの読み込み専用空間で全く提供されません。

4.6.2. 消去

読み込み専用空間のために当てはまりません。

4.6.3. 制限

この空間はﾋﾞｯﾄ ｱﾄﾞﾚｽ指定ではなく、未実装のﾚｼﾞｽﾀは$FFが返ります。

4.6.4. ﾃﾞﾊﾞｲｽ ﾚｼﾞｽﾀ説明

情報については適切なﾃﾞｰﾀｼｰﾄを参照してください。

4.6.4.1. CANBT1～3ﾚｼﾞｽﾀ

CANBT1～3ﾚｼﾞｽﾀはｱﾄﾞﾚｽ$E2～$E4です。

これらは自動ﾋﾞｯﾄ速度を禁止する(EB≠$FF)前に読んでﾌﾞｰﾄ ﾛｰﾀﾞ構成設定空間の"BTC1","BTC2","BTC3"内に再複写することができます(9頁の「ﾋﾞｯﾄ ﾀｲﾐﾝｸﾞ制御1～3 - "BTC1～3"」をご覧ください)。その後、(EB≠$FF中に)常にこのﾋﾞｯﾄ ﾀｲﾐﾝｸﾞで始まるﾌﾞｰﾄ ﾛｰﾀﾞはIAPの場合に非常に有用です。

5. CAN規約とISP命令

本章はCAN網上のより高位の規約と提携するISP命令の符号化を記述します。

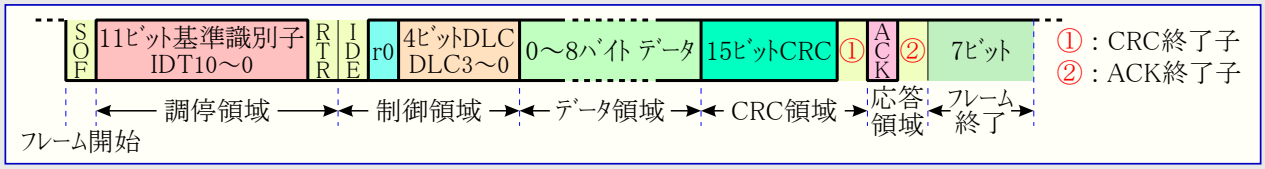
5.1. CANフレーム説明

CAN規約は11ﾋﾞｯﾄ識別子を持つCAN 2.0Aとしても知られるCAN標準ﾌﾚｰﾑ(高速用のISO11898と低速用のSIO11519-2参照)を支援します。

"ﾌﾚｰﾑ開始(SOF"で始まるCAN標準ﾌﾚｰﾑ形式内のﾒｯｾｰｼﾞは、ﾃﾞｰﾀ ﾌﾚｰﾑと遠隔ﾌﾚｰﾑと呼ばれるﾃﾞｰﾀ要求ﾌﾚｰﾑとを区別するのに使われる"遠隔送信要求(RTR)"ﾋﾞｯﾄと識別子から成る"調停領域"が後続します。後続する"制御領域"は"識別子拡張(IDE)"ﾋﾞｯﾄと、"ﾃﾞｰﾀ領域"内の後続するﾃﾞｰﾀ ﾊﾞｲﾄ数を示すのに使われる"ﾃﾞｰﾀ長符号(DLC)"を含みます。遠隔ﾌﾚｰﾑではDLCが要求するﾃﾞｰﾀ ﾊﾞｲﾄ数を含みます。後続する"ﾃﾞｰﾀ領域"は8つまでのﾃﾞｰﾀ ﾊﾞｲﾄを保持することができます。ﾌﾚｰﾑの完全性は後続する"巡回冗長検査(CRC)"和によって保証されます。"応答(ACK)領域"はACK間隔とACK終了子で折衷にします。ACK間隔内のﾋﾞｯﾄは劣性ﾋﾞｯﾄとして送られ、この時に正しく受信したﾃﾞｰﾀを持つ受信側によって優性ﾋﾞｯﾄとして上書きされます。

ISP CAN規約はCAN標準ﾃﾞｰﾀ ﾌﾚｰﾑでだけ使います。

図5-1. CAN標準ﾃﾞｰﾀ ﾌﾚｰﾑ



ISP CAN規約を記述するために、識別子に抽象名が使われますが、既定値は以下の表現内で与えられます。

表5-1. ISP CAN命令の雛形

識別子 (11ﾋﾞｯﾄ)	長さ (4ﾋﾞｯﾄ)	ﾃﾞｰﾀ[0] (1ﾊﾞｲﾄ) ～ ﾃﾞｰﾀ[n-1] (1ﾊﾞｲﾄ)	説明
SYMBLIC_NAME ("CRIS"<<4)+X	n (≦8)	値または意味	命令説明

点対点接続のため、送信CANﾒｯｾｰｼﾞは受信側によってﾊｰﾄﾞｳｪｱ応用が行われるまで繰り返されます。

ﾌﾞｰﾄ ﾛｰﾀﾞは構成設定が見つけられた場合にだけやって来るCANﾌﾚｰﾑに応答することができます。

この機能は多数のCAN節点(ﾉｰﾄﾞ)を持つ網で保証されません。

5.2. CAN ISP命令ﾃﾞｰﾀ列規約

5.2.1. CAN ISP命令説明

この規約を管理するために多数のCANﾒｯｾｰｼﾞ識別子が定義されています。

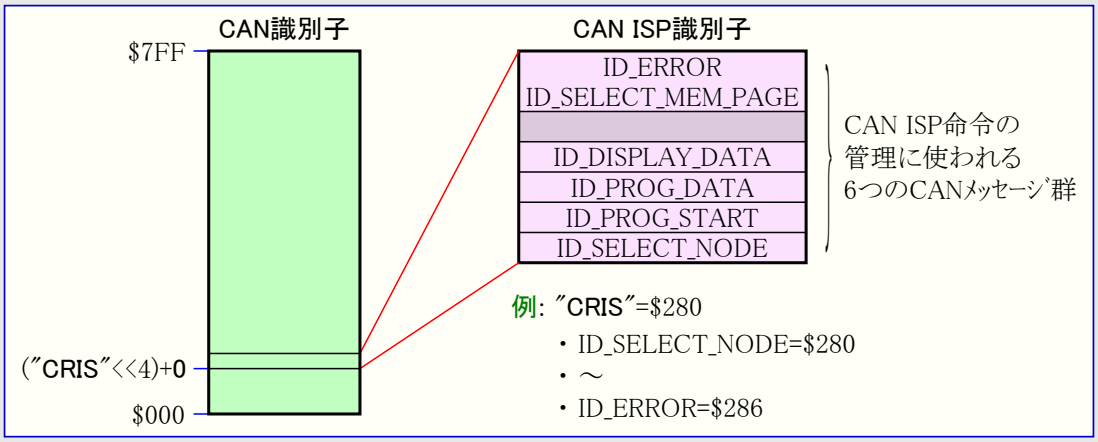
表5-2. CAN ISP規約用に定義されたCANﾒｯｾｰｼﾞ識別子

識別子	ISP命令詳細	値
ID_SELECT_NODE	節点(ﾉｰﾄﾞ)との通信路を開く/閉じる	("CRIS"<<4)+0
ID_PROG_START	ﾒﾓﾘ空間のﾌﾟﾛｸﾞﾗﾐﾝｸﾞ開始	("CRIS"<<4)+1
ID_PROG_DATA	ﾒﾓﾘ空間にﾃﾞｰﾀ書き込み	("CRIS"<<4)+2
ID_DISPLAY_DATA	ﾒﾓﾘ空間からﾃﾞｰﾀ読み込み	("CRIS"<<4)+3
ID_START_APPLI	応用開始	("CRIS"<<4)+4
ID_SELECT_MEM_PAGE	ﾒﾓﾘ空間またはﾍﾟｰｼﾞの選択	("CRIS"<<4)+6
ID_ERROR	ﾌﾞｰﾄ ﾛｰﾀﾞからの異常ﾒｯｾｰｼﾞのみ	

識別子群に関する基礎値を持つ"CRIS"ﾊﾞｲﾄを書くことによってCAN ISP識別子に新しい値を割り当てることが可能です。

最大"CRIS"値は$7Fで既定値は$00です。

図5-2. CAN ISP規約に対するCANﾒｯｾｰｼﾞ識別子の再配置

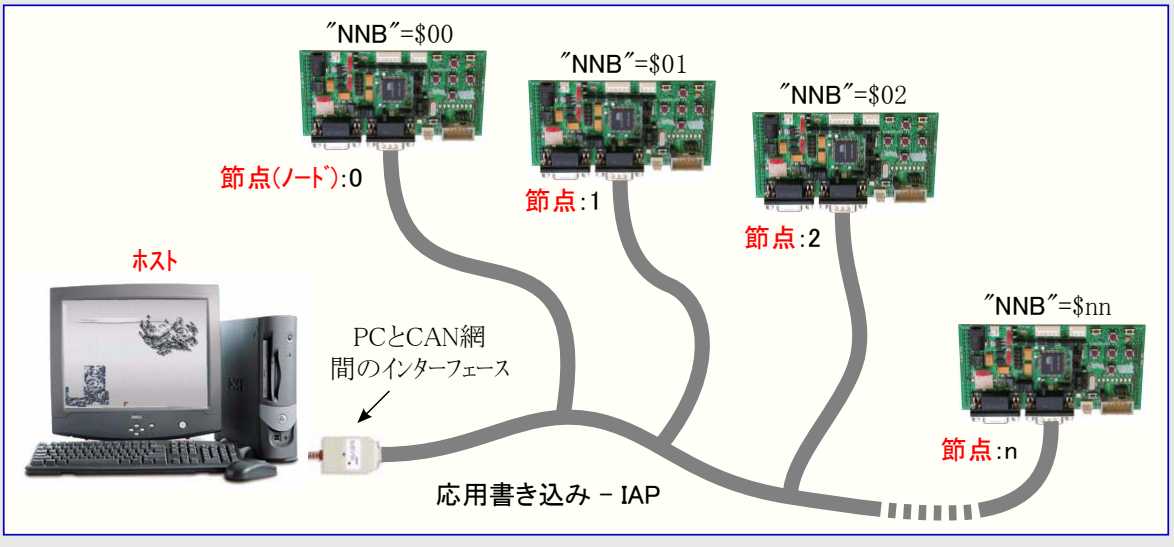


5.2.2. 通信初期化

どんなISP通信を始めるにも先立って装置(CAN節点(ﾉｰﾄﾞ))での通信が開かれなければなりません。装置で通信を開くには、ﾎｽﾄがﾊﾟﾗﾒｰﾀとして渡される節点番号"NNB"を持つ"接続する"CANﾒｯｾｰｼﾞ("ID_SELECT_NODE")を送ります。渡された接点番号が$FFの場合、CANﾌﾞｰﾄ ﾛｰﾀﾞが通信を受け入れます(図5-3.)。さもなければﾊﾟﾗﾒｰﾀで渡された接点番号は局所"NNB"と等しくなければなりません(図5-4.)。

図5-3. CANﾌﾞｰﾄ ﾛｰﾀﾞ初回接続

図5-4. CANﾌﾞｰﾄ ﾛｰﾀﾞ網接続



別の装置で新しい通信を開く前に、現在の装置の通信がそれの接続CANﾒｯｾｰｼﾞ("ID_SELEC_NODE")で閉じられなければなりません。

5.3. CAN ISP命令

5.3.1. CAN節点選択

CAN節点(ﾉｰﾄﾞ)は作業の始めで開かれて終わりで閉じられなければなりません。

5.3.1.1. ﾎｽﾄからのCAN節点選択要求

表5-3. ﾎｽﾄからのCAN節点選択要求

識別子	長さ	ﾃﾞｰﾀ[0]	説明
ID_SELECT_NODE ("CRIS"<<4)+0	1	節点番号("NNB")	指定節点との通信を開くまたは閉じる

5.3.1.2. ﾌﾞｰﾄ ﾛｰﾀﾞからのCAN節点選択応答

表5-4. ﾌﾞｰﾄ ﾛｰﾀﾞからのCAN節点選択応答

識別子	長さ	ﾃﾞｰﾀ[0]	ﾃﾞｰﾀ[1]	説明
ID_SELECT_NODE ("CRIS"<<4)+0	2	"ﾌﾞｰﾄ ﾛｰﾀﾞ改訂"	$00	通信を閉じました。
			$01	通信を開きました。

5.3.2. ﾒﾓﾘ/ﾍﾟｰｼﾞ変更

ﾒﾓﾘ空間と/またはﾍﾟｰｼﾞの変更は1つの命令だけがあり、切り替えはCANﾌﾚｰﾑの"ﾃﾞｰﾀ[0]"によって行われます。

5.3.2.1. ﾎｽﾄからのﾒﾓﾘ/ﾍﾟｰｼﾞ変更要求

表5-5. ﾎｽﾄからのﾒﾓﾘ/ﾍﾟｰｼﾞ変更要求

識別子	長さ	ﾃﾞｰﾀ[0]	ﾃﾞｰﾀ[1]	ﾃﾞｰﾀ[2]	説明
ID_SELECT_MEM_PAGE ("CRIS"<<4)+6	3	$00	ﾒﾓﾘ空間	ﾍﾟｰｼﾞ	活動なし
		$01			ﾒﾓﾘ空間選択
		$02			ﾍﾟｰｼﾞ選択
		$03			ﾒﾓﾘ空間とﾍﾟｰｼﾞを選択

5.3.2.2. ﾌﾞｰﾄ ﾛｰﾀﾞからのﾒﾓﾘ/ﾍﾟｰｼﾞ変更応答

表5-6. ﾌﾞｰﾄ ﾛｰﾀﾞからのﾒﾓﾘ/ﾍﾟｰｼﾞ変更応答

識別子	長さ	ﾃﾞｰﾀ[0]	説明
ID_SELECT_MEM_PAGE ("CRIS"<<4)+6	1	$00	選択OK (例え要求ﾌﾚｰﾑで"ﾃﾞｰﾀ[0]"=$00でも)

5.3.3. ﾒﾓﾘの読み込み/空白検査

これらの操作は通信に於いて直前に開いた装置でだけ実行されます。この命令は直前に定義されたﾒﾓﾘ空間とﾍﾟｰｼﾞで利用可能です。

読み込みまたは空白検査を開始するには、ﾎｽﾄが"ﾃﾞｰﾀ[0]"で望む操作、ﾊﾟﾗﾒｰﾀとして開始ｱﾄﾞﾚｽと終了ｱﾄﾞﾚｽを持つCANﾒｯｾｰｼﾞ("ID_DISPLAY_DATA")を送ります。

5.3.3.1. ﾎｽﾄからのﾒﾓﾘの読み込み/空白検査要求

表5-7. ﾎｽﾄからのﾒﾓﾘの読み込み/空白検査要求

識別子	長さ	ﾃﾞｰﾀ[0]	ﾃﾞｰﾀ[1] ﾃﾞｰﾀ[2]	ﾃﾞｰﾀ[3] ﾃﾞｰﾀ[4]	説明
ID_DISPLAY_DATA ("CRIS"<<4)+3	5	$00	開始ｱﾄﾞﾚｽ (上位,下位)	終了ｱﾄﾞﾚｽ (上位,下位)	選択したﾒﾓﾘ/ﾍﾟｰｼﾞのﾃﾞｰﾀ表示
		$80			選択したﾒﾓﾘ/ﾍﾟｰｼﾞの空白検査

5.3.3.2. ブート ローダからの読み込み/空白検査応答

表5-8. ブート ローダからのメモリの読み込み/空白検査応答

識別子	長さ	データ[0]	データ[1]	～	データ[7]	説明
ID_DISPLAY_DATA ("CRIS"<<4)+3	最大8	8バイトまでのデータ バイト				データ読み込み
	0	–	–	–	–	空白検査OK
	2	非空白の最初のアドレス		–	–	空白検査で誤り
ID_ERROR ("CRIS"<<4)+6	1	$00	–	–	–	ソフトウェア保護設定異常("データ表示"のみ)

5.3.4. メモリの書き込み/消去

これらの操作は通信に於いて直前に開いた装置でだけ実行されます。これらは次の2つの段階が必要です。

・最初の段階は書き込みまたは消去の命令に関するアドレス範囲を指示することです。

・2つ目の段階は書き込みだけに関してデータを送ることです。

書き込み操作を開始するには、ホストが"データ[0]"で望む操作、パラメータとして開始アドレスと終了アドレスを持つ"書き込み開始"CANメッセージ("ID_PROG_START")を送ります。

5.3.4.1. ホストからのメモリの書き込み/消去要求

表5-9. ホストからのメモリの書き込み/消去要求

識別子	長さ	データ[0]	データ[1]	データ[2]	データ[3]	データ[4]	データ[5～7]	説明
ID_PROG_START ("CRIS"<<4)+1	5	$00	開始アドレス (上位,下位)		終了アドレス (上位,下位)		–	選択したメモリ空間/ページ 書き込み開始
	3	$80	$FF	$FF	–	–	–	選択したメモリ空間/ページ消去
ID_PROG_DATA ("CRIS"<<4)+2	最大8	8バイトまでのデータ バイト						書き込むデータ

5.3.4.2. ブート ローダからのメモリの書き込み/消去応答

表5-10. ブート ローダからのメモリの書き込み/消去応答

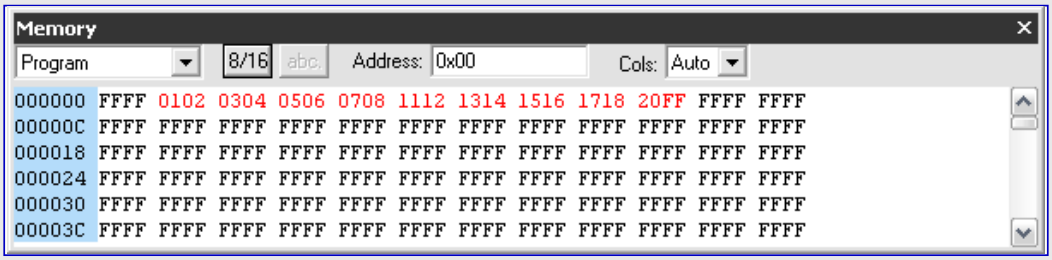
識別子	長さ	データ[0]	説明
ID_PROG_START ("CRIS"<<4)+1	0	–	命令OK
ID_PROG_DATA ("CRIS"<<4)+2	1	$00	命令OK、転送終了
		$02	命令OK、けれども新しい(他の)データを期待
ID_ERROR ("CRIS"<<4)+6	1	$00	異常 – ソフトウェア保護設定 ("書き込み開始"のみ)

5.3.4.3. メモリ書き込み例

表5-11. メモリ書き込み例

要求/応用	CANメッセージ (16進)			説明
	識別子	長さ	データ[0～7]	
要求(→)	000	1	FF	CAN節点(ノード)選択
応答(←)	000	2	03 01	通信を開く
既定メモリ空間=フラッシュメモリ、既定ページ=0ページ				
要求(→)	001	5	00 00 02 00 12	$0002～$0012番地に書き込み
応答(←)	001	0	–	命令OK
要求(→)	002	8	01 02 03 04 05 06 07 08	第1データ列送信
応答(←)	002	1	02	命令OK、新しいデータを期待
要求(→)	002	8	11 12 13 14 15 16 17 18	第2データ列送信
応答(←)	002	1	02	命令OK、新しいデータを期待
要求(→)	002	1	20	第3データ列送信
応答(←)	002	1	00	命令OK、転送終了

図5-5. 直前のﾒﾓﾘ書き込み例の結果



注: AVR Studio®のﾌﾟﾛｸﾞﾗﾑ用ﾒﾓﾘ表示

5.3.5. 応用開始

この操作は通信で直前に開いた装置でだけ実行することができます。

5.3.5.1. ﾎｽﾄからの応用開始要求

応用を開始するには、ﾎｽﾄが"ﾃﾞｰﾀ[1]"で選択する"方法"を持つ応用開始CANﾒｯｾｰｼﾞを送ります。応用はｳｫｯﾁﾄﾞｯｸﾞ ﾘｾｯﾄまたは定義された語ｱﾄﾞﾚｽに飛ぶことによって開始することができます。飛ぶ語ｱﾄﾞﾚｽはSA_H:SA_L(ﾌﾞｰﾄ ﾛｰﾀﾞ構成設定空間)から参照することができます。

表5-12. ﾎｽﾄからの応用開始要求

識別子	長さ	ﾃﾞｰﾀ[0]	ﾃﾞｰﾀ[1]	ﾃﾞｰﾀ[2]	ﾃﾞｰﾀ[3]	説明
ID_START_APPLI ("CRIS"<<4)+4	2	$03	$00	-	-	ｳｫｯﾁﾄﾞｯｸﾞ ﾘｾｯﾄで応用開始
	4		$01	飛ぶ語ｱﾄﾞﾚｽ (上位,下位)		ﾘｾｯﾄなしで語ｱﾄﾞﾚｽ(上位:下位)から応用開始

5.3.5.2. ﾌﾞｰﾄ ﾛｰﾀﾞからの応用開始応答

ﾌﾞｰﾄ ﾛｰﾀﾞによって応答は全く返されません。

6. API - 応用ﾌﾟﾛｸﾞﾗﾑ ｲﾝﾀｰﾌｪｰｽ

6.1. API定義

応用ﾌﾟﾛｸﾞﾗﾑ ｲﾝﾀｰﾌｪｰｽ(API)はｺﾝﾋﾟｭｰﾀ ﾌﾟﾛｸﾞﾗﾑによってそれを行うｻｰﾋﾞｽに対する要求を支援するために、ｺﾝﾋﾟｭｰﾀ ｼｽﾃﾑまたはﾌﾟﾛｸﾞﾗﾑが提供するｿｰｽ ｺｰﾄﾞ ｲﾝﾀｰﾌｪｰｽです。

6.2. API実装

Atmel®のAVR 8ﾋﾞｯﾄ ﾏｲｸﾛ ﾌﾟﾛｾｯｻの特異性はﾌﾗｯｼｭ ﾒﾓﾘを書くことを提供するｺｰﾄﾞがﾌﾞｰﾄ ﾛｰﾀﾞ領域に配置されることを必要とすることです。ﾃﾞﾊﾞｲｽに"減量"CANﾌﾞｰﾄ ﾛｰﾀﾞが書かれている場合、ﾌﾞｰﾄ ﾛｰﾀﾞ領域は既に占有され、使用者用に利用できません。

ﾌﾗｯｼｭ ﾒﾓﾘでの書き込みを使用者に許す解決策はこの"減量"CANﾌﾞｰﾄ ﾛｰﾀﾞに含まれる"flash_wr_block()"ﾙｰﾁﾝを開くことです。その後に使用者は自身の応用からそれを呼び出す(そしてよく吟味された部分ｺｰﾄﾞを使う)ことができます。

6.3. API - 使用の制限

CANﾌﾞｰﾄ ﾛｰﾀﾞはIAR™で開発、ｺﾝﾊﾟｲﾙされました。IARでｺﾝﾊﾟｲﾙされた使用者ﾌﾟﾛｸﾞﾗﾑだけが"減量"CANﾌﾞｰﾄ ﾛｰﾀﾞのAPIにｱｸｾｽすることができます。

6.4. API詳細

6.4.1. 関数名

flash_wr_block()

注: この関数は"flash_boot_lib.c"ﾌｧｲﾙに配置されています。

6.4.2. 特徴

この関数はﾌﾗｯｼｭ ﾒﾓﾘ内で65535ﾊﾞｲﾄ(64Kﾊﾞｲﾄ-1ﾊﾞｲﾄ)までの書き込みを許します。この関数は整列の問題(ﾊﾞｲﾄとﾌﾗｯｼｭ ﾍﾟｰｼﾞの整列)を管理します。

注: 1. この関数は最大64Kﾊﾞｲﾄの供給元緩衝部をﾃﾞﾊﾞｲｽで得ることができないため、1度で65535ﾊﾞｲﾄ全部をｱﾄﾞﾚｽ指定することができません。

2. 64Kより大きなﾌﾗｯｼｭ ﾒﾓﾘ量に関しては、先にﾍﾟｰｼﾞ設定が行われなければなりません(例:RAMPZﾚｼﾞｽﾀの設定または解除)。既定の設定は0です。

6.4.3. 警告

異なったﾍﾟｰｼﾞで書かれるﾊﾞｲﾄはその"flash_wr_block()"関数でなければなりません。

6.4.4. 関数ﾊﾟﾗﾒｰﾀ

1. *src : unsigned charでの位置指示子 – (SRAM内の)供給元緩衝部
2. dest : unsigned short – 目的地、ﾃﾞｰﾀが書かれなければならないﾌﾗｯｼｭ ﾒﾓﾘ内の開始ｱﾄﾞﾚｽ値
3. byte_nb : unsigned short – 書くﾊﾞｲﾄ数

6.4.5. 関数戻り値

(なし)

6.5. 入口点

使用者応用と"減量"CANﾌﾞｰﾄ ﾛｰﾀﾞが一緒にｺﾝﾊﾟｲﾙされないため、入口点が使われます。ﾃﾞﾊﾞｲｽ部品番号に拘らず、入口点は以下のように固定化されます。

```
API_ENTRY_POINT = ( FLASH_SIZE – FLASH_BOOT_SIZE(注) ) + 4ﾊﾞｲﾄ
注: FLASH_BOOT_SIZE=4Kﾊﾞｲﾄ
```

下表は使用ﾃﾞﾊﾞｲｽに対するAPI入口点ｱﾄﾞﾚｽを与えます。

表6-1. API入口点 対 ﾃﾞﾊﾞｲｽ部品番号

部品番号	API_ENTRY_POINT – 語ｱﾄﾞﾚｽ	API_ENTRY_POINT – ﾊﾞｲﾄ ｱﾄﾞﾚｽ
ATmega16M1	$1802	$3004
AT90CAN32, ATmega32M1, ATmega32C1	$3802	$7004
AT90CAN64, ATmega64M1, ATmega64C1	$7802	$F004
AT90CAN128	$0F802	$1F004

6.6. IARでのAPI呼び出し例

使用者ﾌﾟﾛｸﾞﾗﾑは以下のｺｰﾄﾞ(注)を用いてAPIを呼び出すことができます。(注: IAR Cｺﾝﾊﾟｲﾗ用にだけ利用可能)

・ﾄﾞﾗｲﾊﾞ(*.c)ﾌｧｲﾙでの – 関数

```
void (*flash_write) (unsigned char* src, unsigned short dest, unsigned short byte_nb) ¥
      = (void (*)(unsigned char*, unsigned short, unsigned short)) (API_ENTRY_POINT) ;
```

注: ここでENTRY_POINTはﾊﾞｲﾄ ｱﾄﾞﾚｽです。

・ﾍｯﾀﾞ(*.h)ﾌｧｲﾙでの – 関数原型

```
extern void (*flash_write) (unsigned char* src, unsigned short dest, unsigned short byte_nb);
```

6.7. 他のCｺﾝﾊﾟｲﾗでのAPI呼び出し例

6.7.1. IAR Cｺﾝﾊﾟｲﾗと他のｺﾝﾊﾟｲﾗまたはｱｾﾝﾌﾞﾘ言語間の変数渡し

AVRに対してIAR Cｺﾝﾊﾟｲﾗが使われる時にﾚｼﾞｽﾀ ﾌｧｲﾙは図6-1.で示されるように区分されます。

・雑用ﾚｼﾞｽﾀは関数呼び出しを渡って保護されません。
・局所ﾚｼﾞｽﾀは関数呼び出しを渡って保護されます。
・Yﾚｼﾞｽﾀ(R29:R28)はSRAMへのﾃﾞｰﾀ ｽﾀｯｸ ﾎﾟｲﾝﾀとして使われます。
・関数間のﾊﾟﾗﾒｰﾀ渡しと戻り値に雑用ﾚｼﾞｽﾀが使われます。

関数が呼ばれる時に関数に渡されるべきﾊﾟﾗﾒｰﾀはﾚｼﾞｽﾀ ﾌｧｲﾙのR16～R23に置かれます。関数が値を返す時に、ﾊﾟﾗﾒｰﾀと返される値の大きさに依存して、その値はﾚｼﾞｽﾀ ﾌｧｲﾙのR16～R23に置かれます。

表6-2.は関数呼び出し時のﾊﾟﾗﾒｰﾀの配置例を示します。

表6-2. C関数へのﾊﾟﾗﾒｰﾀと配置

関数	ﾊﾟﾗﾒｰﾀ1 ﾚｼﾞｽﾀ	ﾊﾟﾗﾒｰﾀ2 ﾚｼﾞｽﾀ
func(char,char)	R16	R20
func(char,short)	R16	R20,R21
func(short,long)	R16,R17	R20,R21,R22,R23
func(long,long)	R16,R17,18,19	R20,R21,R22,R23

支援されるﾃﾞｰﾀ形式と対応する大きさの完全な参照についてはIAR Systemsからの AVR® IARｺﾝﾊﾟｲﾗ手引きのﾃﾞｰﾀ表現章をご覧ください。

図6-1. ﾚｼﾞｽﾀ ﾌｧｲﾙの区分

6.7.2. "flash_wr_block()"APIで使われるレジスタ

・関数パラメータ
- *src : R17:R16その後に⇒R9:R8、そしてZ(R31:R30)に転送されます。
- dest : R19:R18その後に⇒R25:R24⇒R27:R26(アドレス)
- nb_bytes : R21:R20その後にR5:R4

・他の資源
- Y(R29:R28)は"データ スタック ポインタ"として使われます。
- Z(R31:R30)は"LPM"と"SPM"で使われます。
- "LPM"を使うR0解除と(R0:R1)は"fill_temp_buffer"で使われます。

・使用レジスタの要約

R0, R1, R4, R5, R6, R7, R8, R9, R10, R11, R16,R17, R18, R19, R20, R21, R24, R25, R26, R27, R28, R29, R30, R31

・"flash_boot_lib.lst"ファイル抜粋

```
  138           //-------------------------------------------------------------------------
  139           #pragma location ="API_FLASH"
¥                                 In segment API_FLASH, align 2, keep-with-next
  140           void flash_wr_block(U8* src, U16 dest, U16 byte_nb)
¥                       flash_wr_block:
  141           {
¥    00000000   92BA                ST       -Y, R11
¥    00000002   92AA                ST       -Y, R10
¥    00000004   929A                ST       -Y, R9
¥    00000006   928A                ST       -Y, R8
¥    00000008   927A                ST       -Y, R7
¥    0000000A   926A                ST       -Y, R6
¥    0000000C   925A                ST       -Y, R5
¥    0000000E   924A                ST       -Y, R4
¥    00000010   93BA                ST       -Y, R27
¥    00000012   93AA                ST       -Y, R26
¥    00000014   939A                ST       -Y, R25
¥    00000016   938A                ST       -Y, R24
¥    00000018                       REQUIRE ?Register_R4_is_cg_reg
¥    00000018                       REQUIRE ?Register_R5_is_cg_reg
¥    00000018                       REQUIRE ?Register_R6_is_cg_reg
¥    00000018                       REQUIRE ?Register_R7_is_cg_reg
¥    00000018                       REQUIRE ?Register_R8_is_cg_reg
¥    00000018                       REQUIRE ?Register_R9_is_cg_reg
¥    00000018                       REQUIRE ?Register_R10_is_cg_reg
¥    00000018                       REQUIRE ?Register_R11_is_cg_reg
¥    00000018   0148                MOVW     R9:R8, R17:R16
¥    0000001A   01C9                MOVW     R25:R24, R19:R18
¥    0000001C   012A                MOVW     R5:R4, R21:R20
  142           U8      save_i_flag;
  143           U16     u16_temp, nb_word;
  144           U16     address;
  145           U16     save_page_addr;
  146
  147               //--- Special for API's ------------------------
  148               //- First of all, disabling the Global Interrupt
  149               save_i_flag = SREG;
¥    0000001E   B6AF                IN       R10, 0x3F
  150               Disable_interrupt();
¥    00000020   94F8                CLI
```

注: より多くの詳細については(生成されているなら(コンパイル任意選択)、)以下の*.lstファイルを参照してください。
- "flash_boot_lib.lst"ファイル
- "flash_boot_drv.lst"ファイル

7. 追補A：#define

7.1. ﾌﾟﾛｾｯｻ定義

```
//---------------- ﾌﾞｰﾄﾛｰﾀﾞ定義 ------------------------------------
#define     BOOT_LOADER_SIZE    0x1000      // ﾊﾞｲﾄでの量:4KB
#define     MAX_FLASH_SIZE_TO_ERASE         ( FLASH_SIZE - ((U32)(BOOT_LOADER_SIZE)) )
//---------------- ﾌﾟﾛｾｯｻ定義 --------------------------------------
#define     XRAM_END            XRAMEND     // "ioxxx.h"で定義
#define     RAM_END             RAMEND      // "ioxxx.h"で定義
#define     E2_END              E2END       // "ioxxx.h"で定義
#define     FLASH_END           FLASHEND    // "ioxxx.h"でﾊﾞｲﾄで定義
#define     FLASH_SIZE          ((U32)(FLASH_END)) + 1    // ﾊﾞｲﾄでの量

// 特別な定義用の切り替え
#if   defined(__AT90CAN128__)               // IARによって__HAS_ELPM__定義
#   define MANUF_ID             0x1E        // Atmel(製造者)
#   define FAMILY_CODE          0x81        // AT90CANxxx系列
#   define PRODUCT_NAME         0x97        // 128Kﾊﾞｲﾄのﾌﾗｯｼｭ ﾒﾓﾘ
#   define PRODUCT_REV          0x00        // 改訂0
#   define FLASH_PAGE_SIZE      256         // ﾊﾞｲﾄでの量
#   define _BOOT_CONF_TYPE_     __farflash  // 64K以上に割り当てられるﾌﾞｰﾄﾛｰﾀﾞ
#   define _RAMPZ_IS_USED_                  // 使用ﾌﾗｯｼｭ ﾒﾓﾘが64Kﾊﾞｲﾄ以上の場合にRAMPZﾚｼﾞｽﾀを使用

#elif defined(__AT90CAN64__)                // IARによって__HAS_ELPM__未定義
#   define MANUF_ID             0x1E        // Atmel(製造者)
#   define FAMILY_CODE          0x81        // AT90CANxxx系列
#   define PRODUCT_NAME         0x96        // 64Kﾊﾞｲﾄのﾌﾗｯｼｭ ﾒﾓﾘ
#   define PRODUCT_REV          0x00        // 改訂0
#   define FLASH_PAGE_SIZE      256         // ﾊﾞｲﾄでの量
#   define _BOOT_CONF_TYPE_     __flash     // 64K以下に割り当てられるﾌﾞｰﾄﾛｰﾀﾞ

#elif defined(__AT90CAN32__)                // IARによって__HAS_ELPM__未定義
#   define MANUF_ID             0x1E        // Atmel(製造者)
#   define FAMILY_CODE          0x81        // AT90CANxxx系列
#   define PRODUCT_NAME         0x95        // 32Kﾊﾞｲﾄのﾌﾗｯｼｭ ﾒﾓﾘ
#   define PRODUCT_REV          0x00        // 改訂0
#   define FLASH_PAGE_SIZE      256         // ﾊﾞｲﾄでの量
#   define _BOOT_CONF_TYPE_     __flash     // 64K以下に割り当てられるﾌﾞｰﾄﾛｰﾀﾞ

#elif defined(__ATmega16M1__)               // IARによって__HAS_ELPM__未定義
#   define MANUF_ID             0x1E        // Atmel(製造者)
#   define FAMILY_CODE          0x84        // ATmegaxxM1/C1系列
#   define PRODUCT_NAME         0x94        // 16Kﾊﾞｲﾄのﾌﾗｯｼｭ ﾒﾓﾘ
#   define PRODUCT_REV          0x00        // 改訂0
#   define FLASH_PAGE_SIZE      128         // ﾊﾞｲﾄでの量
#   define _BOOT_CONF_TYPE_     __flash     // 64K以下に割り当てられるﾌﾞｰﾄﾛｰﾀﾞ

#elif defined(__ATmega32M1__)               // IARによって__HAS_ELPM__未定義
#   define MANUF_ID             0x1E        // Atmel(製造者)
#   define FAMILY_CODE          0x84        // ATmegaxxM1/C1系列
#   define PRODUCT_NAME         0x95        // 32Kﾊﾞｲﾄのﾌﾗｯｼｭ ﾒﾓﾘ
#   define PRODUCT_REV          0x00        // 改訂0
#   define FLASH_PAGE_SIZE      128         // ﾊﾞｲﾄでの量
#   define _BOOT_CONF_TYPE_     __flash     // 64K以下に割り当てられるﾌﾞｰﾄﾛｰﾀﾞ

#elif defined(__ATmega32C1__)               // IARによって__HAS_ELPM__未定義
#   define MANUF_ID             0x1E        // Atmel(製造者)
#   define FAMILY_CODE          0x86        // ATmegaxxM1/C1系列
#   define PRODUCT_NAME         0x95        // 32Kﾊﾞｲﾄのﾌﾗｯｼｭ ﾒﾓﾘ
#   define PRODUCT_REV          0x00        // 改訂0
#   define FLASH_PAGE_SIZE      128         // ﾊﾞｲﾄでの量
#   define _BOOT_CONF_TYPE_     __flash     // 64K以下に割り当てられるﾌﾞｰﾄﾛｰﾀﾞ
```

```
#elif defined(__ATmega32C1__)              // IARによって__HAS_ELPM__未定義
#   define MANUF_ID            0x1E        // Atmel(製造者)
#   define FAMILY_CODE         0x86        // ATmegaxxM1/C1系列
#   define PRODUCT_NAME        0x95        // 32Kﾊﾞｲﾄのﾌﾗｯｼｭ ﾒﾓﾘ
#   define PRODUCT_REV         0x00        // 改訂0
#   define FLASH_PAGE_SIZE     128         // ﾊﾞｲﾄでの量
#   define _BOOT_CONF_TYPE_    __flash     // 64K以下に割り当てられるﾌﾞｰﾄ ﾛｰﾀﾞ

#elif defined(__ATmega64M1__)              // IARによって__HAS_ELPM__未定義
#   define MANUF_ID            0x1E        // Atmel(製造者)
#   define FAMILY_CODE         0x84        // ATmegaxxM1/C1系列
#   define PRODUCT_NAME        0x96        // 64Kﾊﾞｲﾄのﾌﾗｯｼｭ ﾒﾓﾘ
#   define PRODUCT_REV         0x00        // 改訂0
#   define FLASH_PAGE_SIZE     256         // ﾊﾞｲﾄでの量
#   define _BOOT_CONF_TYPE_    __flash     // 64K以下に割り当てられるﾌﾞｰﾄ ﾛｰﾀﾞ

#elif defined(__ATmega64C1__)              // IARによって__HAS_ELPM__未定義
#   define MANUF_ID            0x1E        // Atmel(製造者)
#   define FAMILY_CODE         0x86        // ATmegaxxM1/C1系列
#   define PRODUCT_NAME        0x96        // 64Kﾊﾞｲﾄのﾌﾗｯｼｭ ﾒﾓﾘ
#   define PRODUCT_REV         0x00        // 改訂0
#   define FLASH_PAGE_SIZE     256         // ﾊﾞｲﾄでの量
#   define _BOOT_CONF_TYPE_    __flash     // 64K以下に割り当てられるﾌﾞｰﾄ ﾛｰﾀﾞ

#else
#   error IAR Embedded Workbench IDEでの不正なﾃﾞﾊﾞｲｽ選択
#endif

#define     FOSC                8000        // 8MHz外部ｸﾘｽﾀﾙ
```

7.2. CAN接続定義

```
//---------------- CAN定義 -------------
#define CAN_BAUDRATE  CAN_AUTOBAUD

#if   defined(__AT90CAN128__) || defined(__AT90CAN64__) || defined(__AT90CAN32__)
#   define CAN_PORT_IN    PIND
#   define CAN_PORT_DIR   DDRD
#   define CAN_PORT_OUT   PORTD
#   define CAN_INPUT_PIN  6
#   define CAN_OUTPUT_PIN 5
#   define NB_MOB         15
#elif  defined(__ATmega16M1__) || defined(__ATmega32M1__) || defined(__ATmega32C1__) || ¥
       defined(__ATmega64M1__) || defined(__ATmega64C1__)
#   define CAN_PORT_IN    PINC
#   define CAN_PORT_DIR   DDRC
#   define CAN_PORT_OUT   PORTC
#   define CAN_INPUT_PIN  3
#   define CAN_OUTPUT_PIN 2
#   define NB_MOB         6
#else
#   error IAR Embedded Workbench IDEでの不正なﾃﾞﾊﾞｲｽ選択
#endif
```

7.3. ブート ローダ定義

```
//--------------- ブート ローダ構成設定 --------------
#define BOOT_LOADER_SIZE        0x1000              // バイトでの量:4KB
#define MAX_FLASH_SIZE_TO_ERASE ( FLASH_SIZE - ((U32)(BOOT_LOADER_SIZE)) )

#define BOOT_VERSION            0x03
#define BOOT_ID1                0xD1
#define BOOT_ID2                0xD2

#define BSB_DEFAULT             0xFF
#define SSB_DEFAULT             0xFF
#define EB_DEFAULT              0xFF
#define BTC1_DEFAULT            0xFF
#define BTC2_DEFAULT            0xFF
#define BTC3_DEFAULT            0xFF
#define NNB_DEFAULT             0xFF
#define CRIS_DEFAULT            0x00
#define SA_H_DEFAULT            0x00
#define SA_L_DEFAULT            0x00

#define BSB   ((U16) &boot_conf[0])
#define SSB   ((U16) &boot_conf[1])
#define EB    ((U16) &boot_conf[2])
#define BTC1  ((U16) &boot_conf[3])
#define BTC2  ((U16) &boot_conf[4])
#define BTC3  ((U16) &boot_conf[5])
#define NNB   ((U16) &boot_conf[6])
#define CRIS  ((U16) &boot_conf[7])
#define SA_H  ((U16) &boot_conf[8])
#define SA_L  ((U16) &boot_conf[9])

#define BOOT_CONF_SIZE            10
#define SSB_INDEX               0x01
#define SSB_NO_SECURITY         0xFF
#define SSB_WR_PROTECTION       0xFE
#define SSB_RD_WR_PROTECTION    0xFC
```

7.4. メモリ定義

```
//-------- メモリ定義 -----------------
#define MEM_FLASH               0
#define MEM_EEPROM              1
#define MEM_SIGNATURE           2
#define MEM_BOOT_INF            3                   // ブート ローダ情報
#define MEM_BOOT_CONF           4                   // ブート ローダ構成設定
#define MEM_HW_REG              5
#define MEM_DEF_MAX             MEM_HW_REG
#define MEM_DEFAULT             MEM_FLASH

#define PAGE_DEFAULT            0x00
#define ADD_DEFAULT             0x0000
#define N_DEFAULT               0x0001
```

7.5. CAN規約定義

```
//------ IAPﾃﾞｰﾀ --------------------------
#define MAX_BASE_ISP_IAP_ID     0x7F0
#define MIN_BASE_ISP_IAP_ID     0x000

//------ 規約命令 -----------------
#define CAN_ID_SELECT_NODE      0x00

#define CAN_ID_PROG_START       0x01
#   define CAN_INIT_PROG        0x00
#   define CAN_FULL_ERASE_1     0x80
#   define CAN_FULL_ERASE_2     0xFF
#   define CAN_FULL_ERASE_3     0xFF

#define CAN_ID_PROG_DATA        0x02

#define CAN_ID_DISPLAY_DATA     0x03
#   define CAN_READ_DATA        0x00
#   define CAN_BLANK_CHECK      0x80

#define CAN_ID_START_APPLI      0x04
#   define CAN_START_APPLI      0x03
#   define CAN_RESET_APPLI      0x00
#   define CAN_JUMP_APPLI       0x01

#define CAN_ID_SELECT_MEM_PAGE  0x06
#   define CAN_NO_ACTION        0x00
#   define CAN_SEL_MEM          0x01
#   define CAN_SEL_PAGE         0x02
#   define CAN_SEL_MEM_N_PAGE   0x03

#define CAN_ID_ERROR            0x06

#define COMMAND_OK              0x00
#define OK_END_OF_DATA          0x00
#define OK_NEW_DATA             0x02

#define LOCAL_BUFFER_SIZE       0x100
```

8. 追補B：CAN規約要約

表8-1. CAN規約要約 – ホストからの要求

ISP命令要求識別子	長さ	データ								説明
		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	
ID_SELECT_NODE ("CRIS"<<4)+0	1	節点	–	–	–	–	–	–	–	通信を開くまたは閉じる
ID_PROG_START ("CRIS"<<4)+1	5	$00	開始アドレス (上位,下位)		終了アドレス (上位,下位)		–	–	–	書き込みの初期化
	3	$80	$FF	$FF	–	–	–	–	–	消去
ID_PROG_DATA ("CRIS"<<4)+2	最大8	8バイトまでのデータ バイト								書き込むデータ
ID_DISPLAY_DATA ("CRIS"<<4)+3	5	$00	開始アドレス (上位,下位)		終了アドレス (上位,下位)		–	–	–	データ表示(読み込み)
		$80								空白検査
ID_START_APPLI ("CRIS"<<4)+4	2	$03	$00	–	–	–	–	–	–	リセットで応用開始
	4		$01	$0000						応用開始、0番地へ飛ぶ
ID_SELECT_MEM_PAGE ("CRIS"<<4)+6	3	$00	メモリ空間	ページ	–	–	–	–	–	活動なし
		$01								メモリ空間選択
		$02								ページ選択
		$03								メモリ空間とページを選択

表8-2. CAN規約要約 – ブート ローダからの応答

ISP命令応答識別子	長さ	データ								説明
		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	
ID_SELECT_NODE ("CRIS"<<4)+0	2	ブート ローダ 改訂	$00	–	–	–	–	–	–	通信を閉じる
			$01							通信を開く
ID_PROG_START ("CRIS"<<4)+1	0	–	–	–	–	–	–	–	–	命令OK
ID_PROG_DATA ("CRIS"<<4)+2	1	$00	–	–	–	–	–	–	–	命令OK、転送の最後
		$02								命令OK、新しいデータを期待
ID_DISPLAY_DATA ("CRIS"<<4)+3	最大8	8バイトまでのデータ バイト								データ読み込み
	0	–	–	–	–	–	–	–	–	リセットで応用開始
	2	非空白の最初のアドレス		–	–	–	–	–	–	空白検査での異常
ID_SELECT_MEM_PAGE またはID_ERROR ("CRIS"<<4)+6	1	$00	–	–	–	–	–	–	–	選択OKまたはソフトウェア保護設定異常

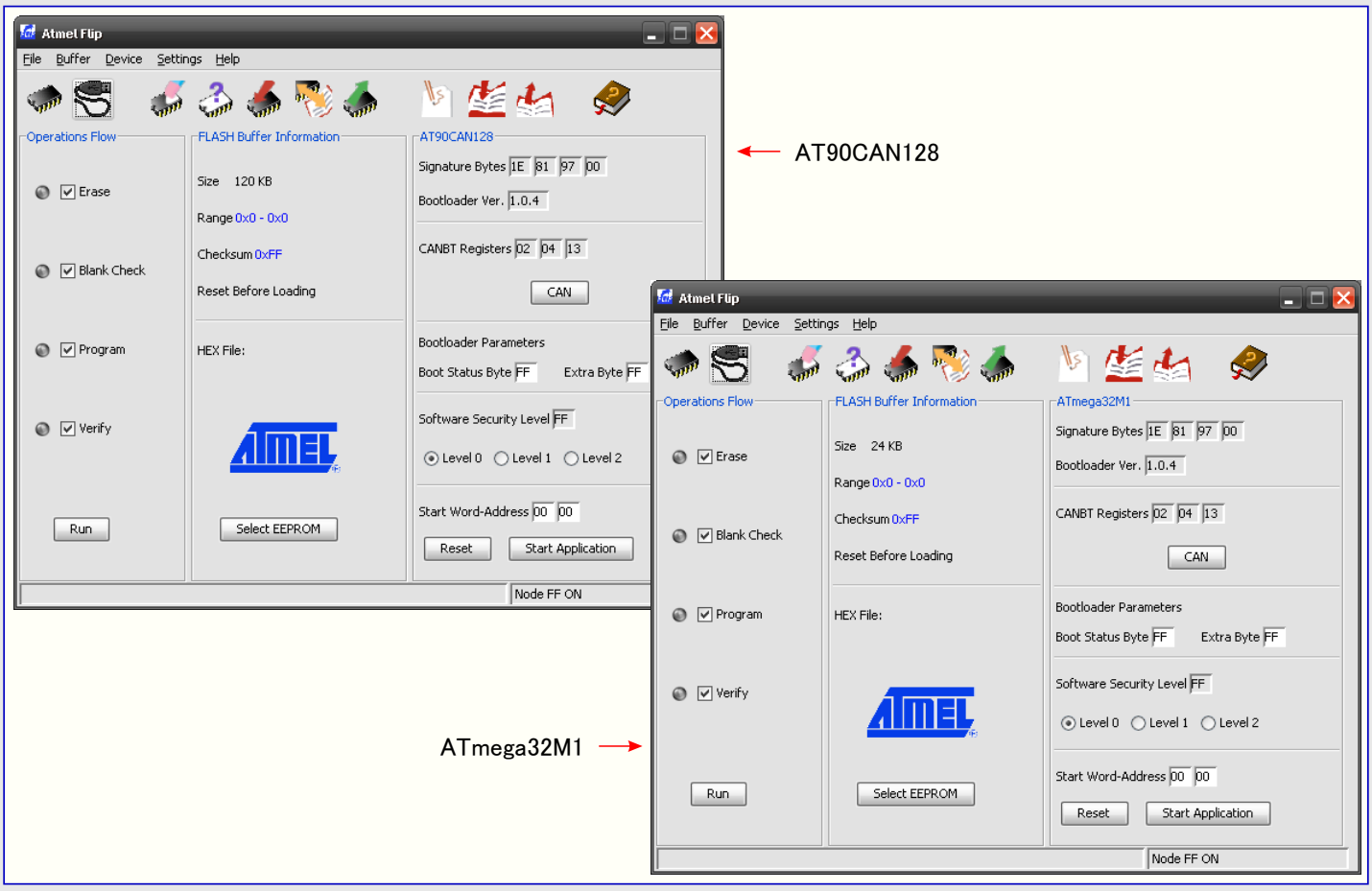
9. 追補C：FLIPインターフェース

FLIPはAtmelのマイクロ コントローラ デバイスを実装書き込みする柔軟なPC応用です。このFLIPの主新版は以下の能力を提供します。

- RS232、USB、CANのインターフェースを通しての実装書き込みの実行
- それの直感的な画像使用者インターフェースまたは、DOS®窓(batchisp手引書をご覧ください)やKEIL's uVision2のような組み込みソフトウェアIDE、更にあなた自身の応用(ISP関数ライブラリ手引書をご覧ください)からの起動を通して使えます。
- Windows® 9x/Me/NT®/2000/XP®下で走行
- データ ファイルの読み込みと保存にIntel® MCS-86 16進オブジェクト、コード88ファイル形式を支援
- 緩衝部編集能力：塗り、検索、複写、書き込み、照合、読み込み、保護レベルと特殊バイトの読み出しと設定
- デバイス逐次化能力(batchispからのみ)
- ISPハードウェア条件がソフトウェアによって設定可能
- どんな目的対象ハードウェアもなしに実演動作形態がISP操作を擬態

FLIPリンク：http://www.atmel.com/dyn/resources/prod_documents/Flip_Installer_3_3_4.exe

図9-1. FLIPウィンドウ例



10. 追補D：目的対象デバイスに対するコンパイルの注意

参照：・AVR用IAR C/C++コンパイラ 5.20.1 (5.20.50092)

・"..¥$PROJ_DIR$¥config.h"ファイルでの基盤選択

10.1. 目的対象選択

Project⇒Options⇒General Options⇒Target⇒Processor Configuration (以下の何れか)

・--cpu=can128, AT90CAN128

・--cpu=can64, AT90CAN64

・--cpu=can32, AT90CAN32

・--cpu=32m1, ATmega32M1

・--cpu=32c1, ATmega32C1

(以下は2009年4月13日現在、未実装)

・--cpu=64m1, ATmega64M1

・--cpu=64c1, ATmega64C1

・--cpu=16m1, ATmega16M1

10.2. 最適化

Project⇒Options⇒C/C++ Compiler⇒Optimizations

・Speed : High(最大最適化)

・Number of cross-call passes : Ulimited

・Always do cross call optimization : ON

10.3. 対応するﾘﾝｶ命令ﾌｧｲﾙの選択

Project⇒Options⇒Linker⇒Config⇒Linker Command File (以下の何れか)

- Override defaultをﾁｪｯｸ
- ..¥$PROJ_DIR$¥can128_iar_can_bootloader_link.xcl
- ..¥$PROJ_DIR$¥can64_iar_can_bootloader_link.xcl
- ..¥$PROJ_DIR$¥can32_iar_can_bootloader_link.xcl
- ..¥$PROJ_DIR$¥m64m1c1_iar_can_bootloader_link.xcl
- ..¥$PROJ_DIR$¥m32m1c1_iar_can_bootloader_link.xcl
- ..¥$PROJ_DIR$¥m16m1c1_iar_can_bootloader_link.xcl

10.4. ｺﾝﾊﾟｲﾙ

1. Project⇒Clean
2. Project⇒Rebuilt All

生成物：″IAR_can_boot_loader.a90″ (″*.hex″と等価)と″IAR_can_boot_loader.dbg″

位置　：..¥$PROJ_DIR$¥output_iar¥debug¥exe¥

10.5. 予めｺﾝﾊﾟｲﾙされたHEXﾌｧｲﾙ

(使われる基盤に応じて)いくつかの予めｺﾝﾊﾟｲﾙされたHEXﾌｧｲﾙが以下に作られています。

..¥$PROJ_DIR$¥output_iar¥debug¥exe¥pre_compiled_hex_file¥

- AT90CAN128用　　：IAR_can_boot_loader_dvk90can1_at90can128.hexまたはIAR_can_boot_loader_stk600_at90can128.hex
- AT90CAN64用　　：IAR_can_boot_loader_stk600_at90can64.hex
- AT90CAN32用　　：IAR_can_boot_loader_stk600_at90can32.hex
- ATmega32M1/C1用：IAR_can_boot_loader_stk600_atmega32m1c1.hexまたはIAR_can_boot_loader_mc310_atmega32m1c1.hex

本社

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
TEL 1(408) 441-0311
FAX 1(408) 487-2600

国外営業拠点

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
TEL (852) 2245-6100
FAX (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
TEL (33) 1-30-60-70-00
FAX (33) 1-30-60-71-11

Atmel Japan
104-0033 東京都中央区
新川1-24-8
東熱新川ﾋﾞﾙ 9F
ｱﾄﾒﾙ ｼﾞｬﾊﾟﾝ株式会社
TEL (81) 03-3523-3551
FAX (81) 03-3523-7581

製品窓口

ｳｪﾌﾞｻｲﾄ
www.atmel.com

技術支援
avr@atmel.com

販売窓口
www.atmel.com/contacts

文献請求
www.atmel.com/literature

本応用記述はAtmelのAVR076応用記述(doc8247.pdf Rev.8247A-08/09)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。