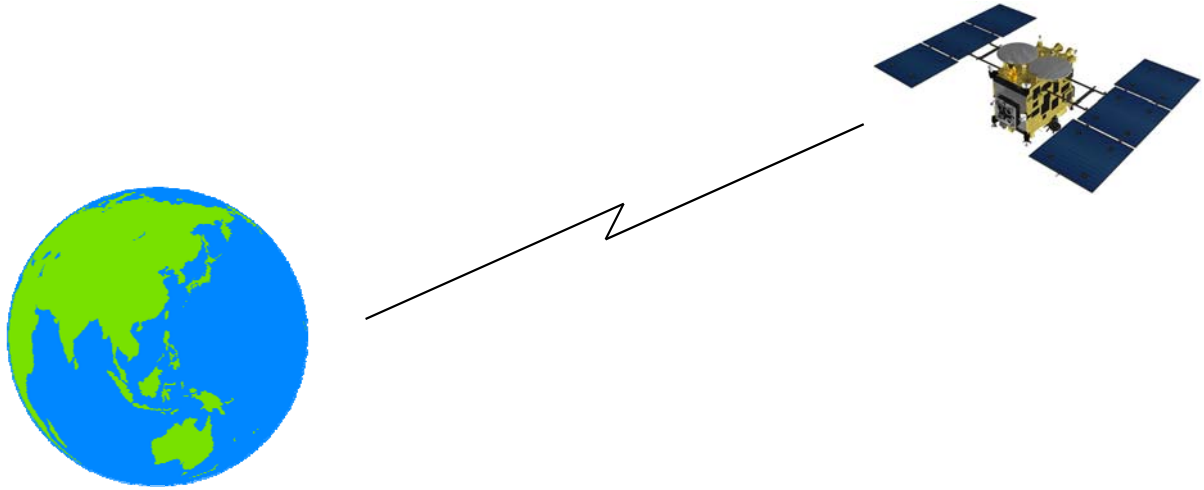
ソフトウェア作成の課題 4
Sレコード変換

v4.1

「はやぶさ」へのプログラム送信、データの受信にはSレコードが使われている

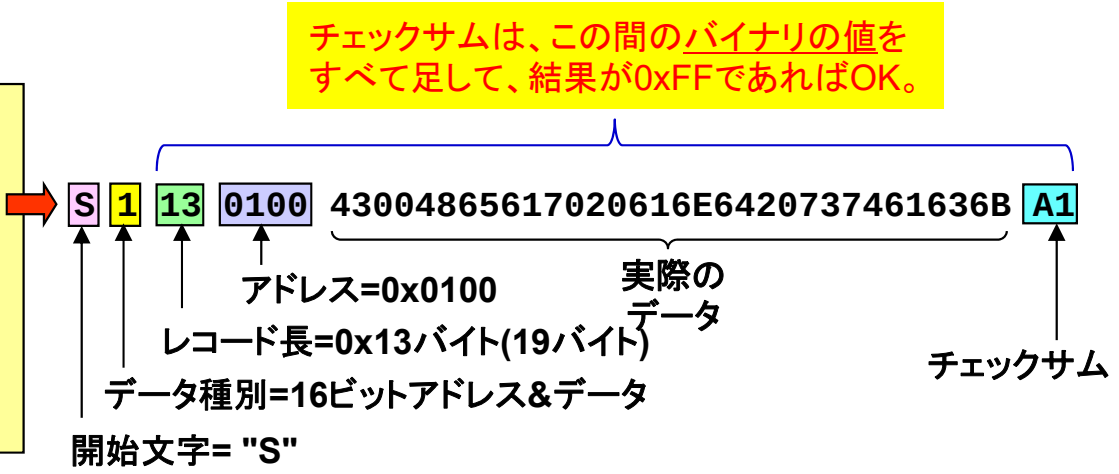


SレコードとHEXフォーマット(16進ACSII文字)

- バイナリデータを16進ASCIIキャラクタに変換して転送するフォーマット。
  - マイコンの初期段階から使われていて、Sレコード形式と、HEX形式2つが有名。
  - 1バイトを2文字の16進ASCII文字で表現するため、効率は悪いが、フォーマットが単純でロード(メモリに配置)するアドレスを指定できるなど、現在でも利用されている。
  - JAXAの「はやぶさ」もこの方式でプログラムなどを送っている(プロトコルが不要)。

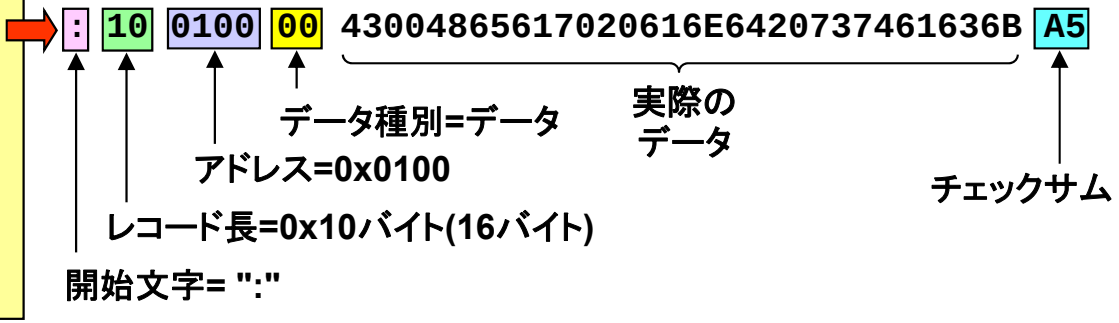
モトローラ Sレコードフォーマットの例

```
S00A000074322E7372656374
S1130100430048656617020616E6420737461636BA1
S113011020636F6C6C6973696F6E0A007907FEFC6B
S113012079001D9479011D9A192269820B801D1092
····(中略)····
S105FEFC0001FF
S903011CDF
```



インテルHEXフォーマットの例

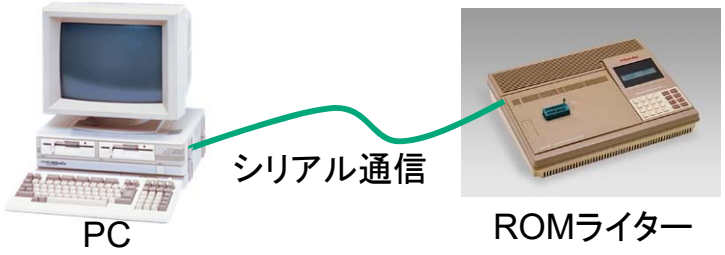
```
:1001000043004865617020616E6420737461636BA5
:1001100020636F6C6C6973696F6E0A007907FEFC6F
:1001200079001D9479011D9A192269820B801D1096
····(中略)····
:02FEFC00000103
:0400000300000011CDC
:00000001FF
```



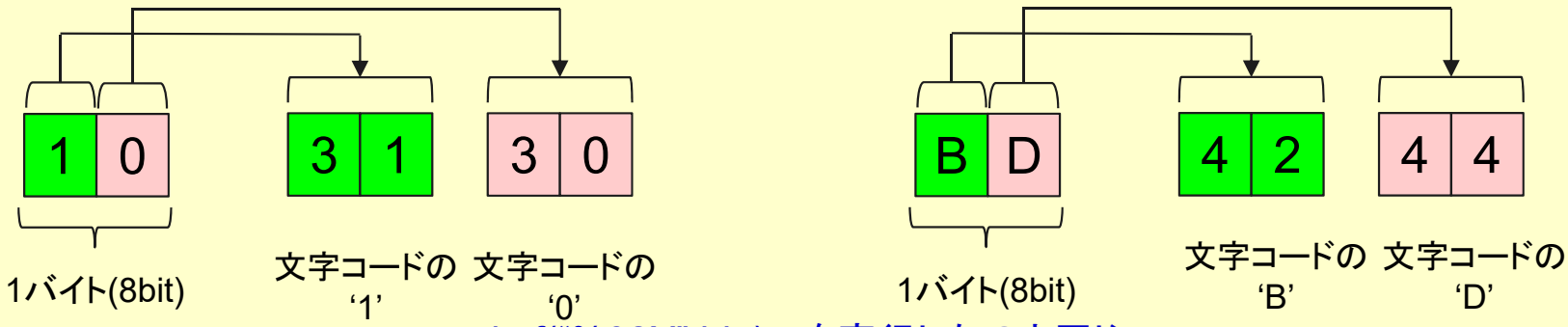
上記は、全く同じデータを2つのフォーマットで表している。

なぜ、SレコードやHEXフォーマットがあるか?

- シリアル通信でバイナリデータを送る場合、データに制御文字(0x00～0x1F)が含まれるために、そのままでは送れない (次ページのASCIIコード表を参照)
  - 制御文字は、本来の制御コードとして扱われるためデータとしては転送されない。
  - 例: 0x0D はリターンコード、0x0Aは改行コードなど。



- そのため、すべてのデータを4ビット毎に16進の文字コードにすることで、バイナリデータの転送を可能とする方式。
  - 1バイトを2バイトの文字にして送るため、実際の転送バイト数は、元のデータの2倍になる。
  - ちなみに、プロトコルを利用してバイナリを送る方法もあります(kermit、Xmodemなど)。



8ビット(8単位)ASCII文字コード表

	上位	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
下位		0x	1x	2x	3x	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
0000	x0	NUL	DLE	SP	0	@	P	`	p				ー	タ	ミ		
0001	x1	SOH	DC1	!	1	A	Q	a	q			。	ア	チ	ム		
0010	x2	STX	DC2	″	2	B	R	b	r			「	イ	ツ	メ		
0011	x3	ETX	DC3	#	3	C	S	c	s			」	ウ	テ	モ		
0100	x4	EOT	DC4	$	4	D	T	d	t			。	エ	ト	ヤ		
0101	x5	ENQ	NAK	%	5	E	U	e	u			・	オ	ナ	ユ		
0110	x6	ACK	SYN	&	6	F	V	f	v			ヲ	カ	ニ	ヨ		
0111	x7	BEL	ETB	’	7	G	W	g	w			ァ	キ	ヌ	ラ		
1000	x8	BS	CAN	(	8	H	X	h	x			ィ	ク	ネ	リ		
1001	x9	HT	EM	)	9	I	Y	i	y			ゥ	ケ	ノ	ル		
1010	xA	LF	EOF	*	:	J	Z	j	z			ェ	コ	ハ	レ		
1011	xB	VT	ESC	+	;	K	[	k	{			ォ	サ	ヒ	ロ		
1100	xC	FF	FS	,	<	L	¥	l	|			ャ	シ	フ	ワ		
1101	xD	CR	GS	-	=	M	]	m	}			ュ	ス	ヘ	ン		
1110	xE	SO	RS	.	>	N	^	n	~			ョ	セ	ホ	゛		
1111	xF	SI	US	/	?	O	_	o	DEL			ッ	ソ	マ	゜		

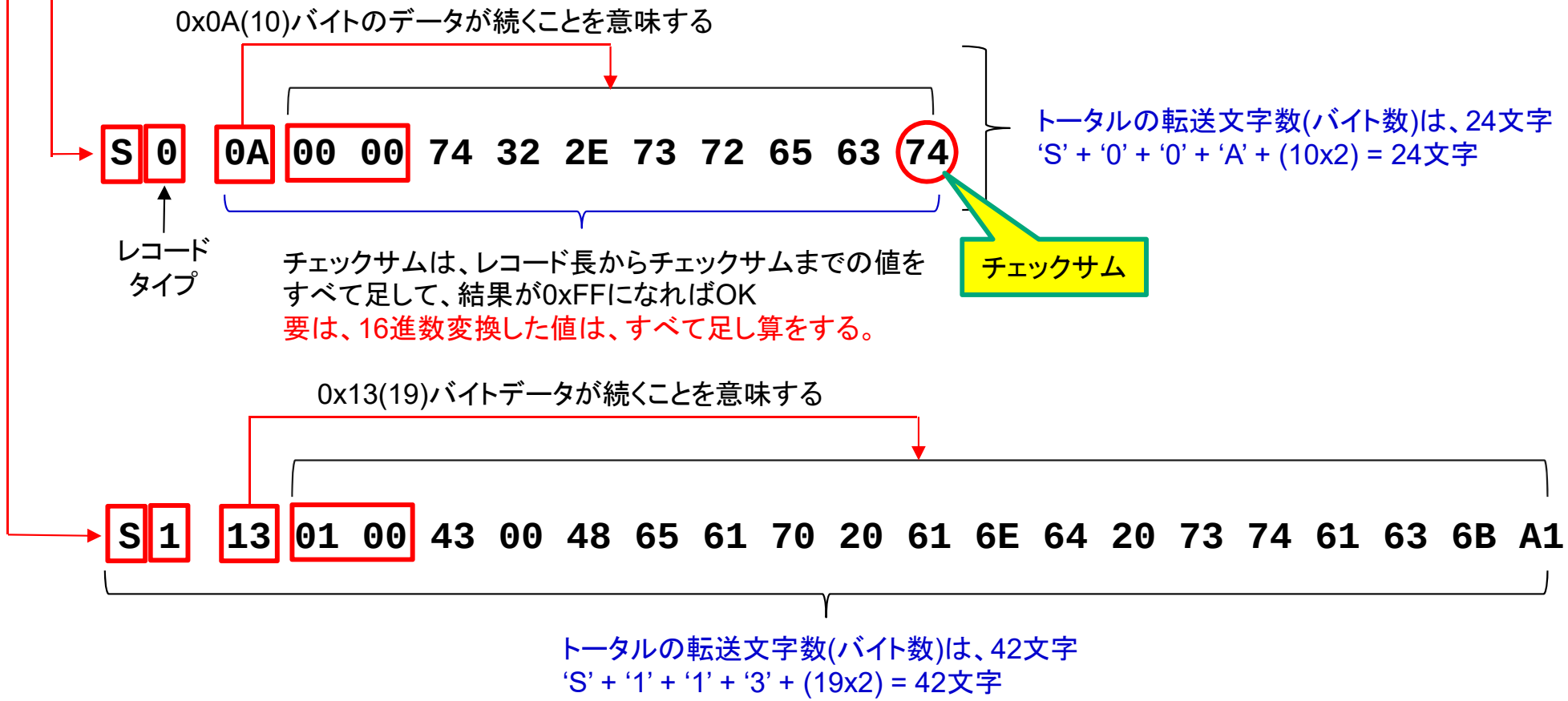
シリアル通信では、この部分のコードを、
文字として転送できない

Shift-JISコードの第1バイト目のコード
0x81～0x9F || 0xE0～0xEF

Sレコードからバイナリにするコマンド
srec2bin (仮称)

実際のSレコードの内容

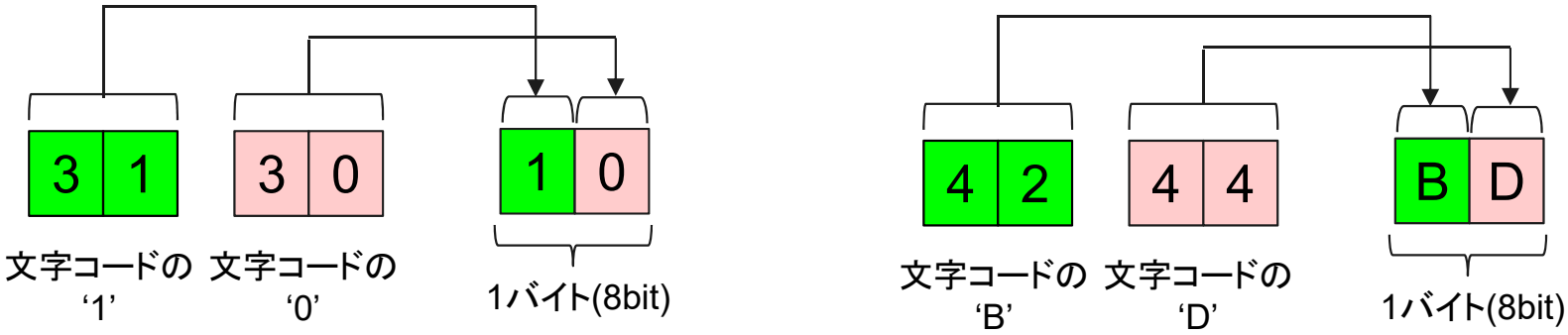
```
S00A000074322E7372656374
S1130100430048656170206E6420737461636BA1
S113011020636F6C6C6973696F6E0A007907FEFC6B
S113012079001D9479011D9A192269820B801D1092
S105FEFC0001FF
S903011CDF
```



16進文字列からバイナリの変換

- Sレコードの文字列からバイナリーの変換は、下記のように行う。
- 2バイトの文字コードから、1バイトのバイナリを作る。
  - 16進変換を方法を理解するために、strtol()関数は使わずに、自分で変換関数を作成する。

- 文字コードから'0' (0x30)を引く
  - 引いた結果が、10より小さければ、0～9の範囲
  - 引いた結果が、10以上であれば、さらに7 ('A' – ':') を引く



変換のヒント

2文字を、上位4ビット(1文字目)、下位4ビット(2文字目)の2回に分けて変換。
最後に、「上位4ビット << 4 | 下位4ビット」を実行して1バイトのデータにする。

Sレコードからバイナリにするコマンドのシンタックス

```
C:¥Users¥m-hoshi>srec2bin/?
構文 : srec2bin [<opts>] [[/i[=]]<inpath>] [[/o[=]]<outpath>] [<opts>]
機能 : Sレコードをバイナリに変換
オプション:
      /i[=]<入力パス名> 入力パス(デフォルト = stdin)
      /o[=]<出力パス名> 出力パス(デフォルト = stdout)
      /r                出力ファイルのリライト指定
      /?                ヘルプ表示
```

/オプションについては、あとで説明をします。

/i は、読み込みをするSレコードのテキストファイルを指定します。

/o は、Sレコードからバイナリに変換した出力ファイルを指定します。

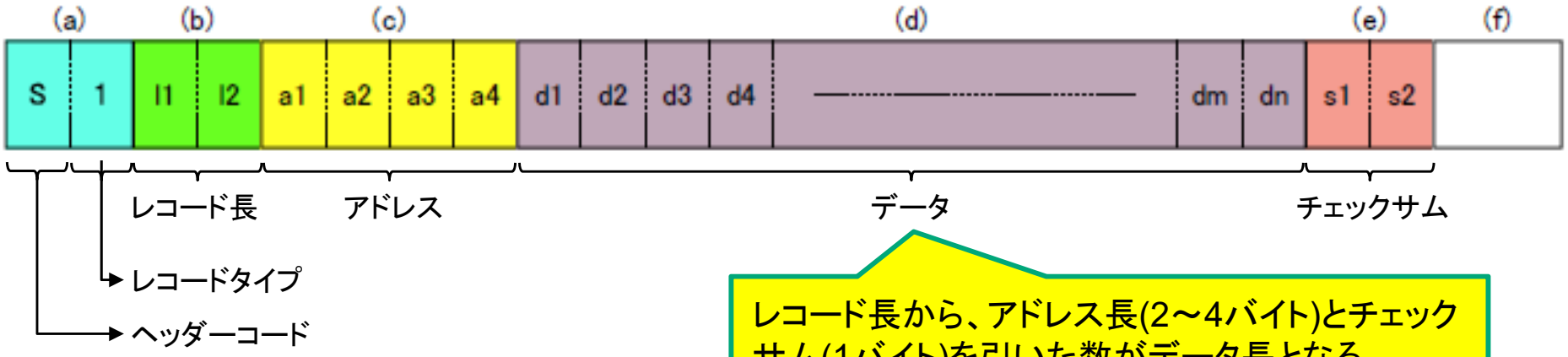
通常、ファイル名の指定はオプションを使わず、コマンド名のあとにファイル名を書くだけで動作します。そして、ファイル名が1つの時は、入力ファイルのみとなり、ファイル名が2つあるときは、最初が入力ファイル、2つ目が出力ファイルにします。ファイル名が3個以上あったら、エラーとします。

/i と /o オプションは、入力したファイル名の順番に関係なく、強制的に入力ファイルと出力ファイルを指定するためのものです。すでにファイル名が入っているかのチェックは必要ありません。/i と /o で指定したファイル名は、入力ファイル、出力ファイルとも上書きをしてください。

Sレコード変換プログラム作成のヒント

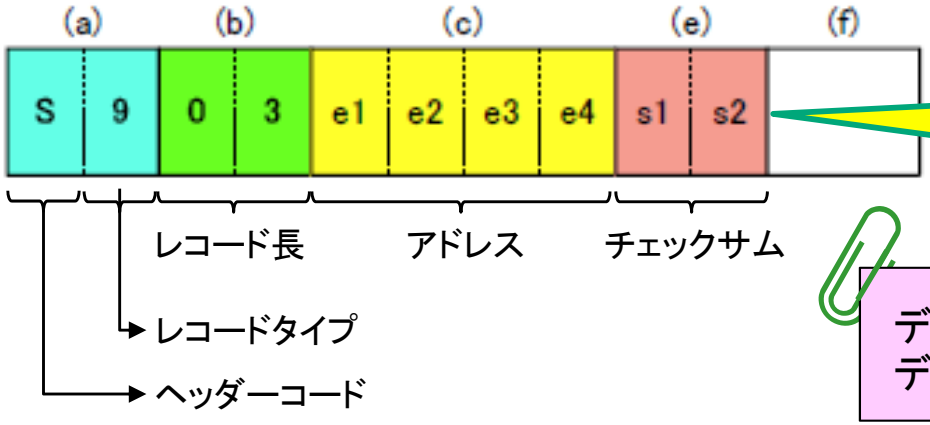
Sレコードのフォーマット

データレコード(S1～S3)



レコード長から、アドレス長(2～4バイト)とチェックサム(1バイト)を引いた数がデータ長となる。

終了レコード(S7～S9)



終了レコードの場合、アドレス長とチェックサム(1バイト)を引くとデータ長は0(ゼロ)になる。

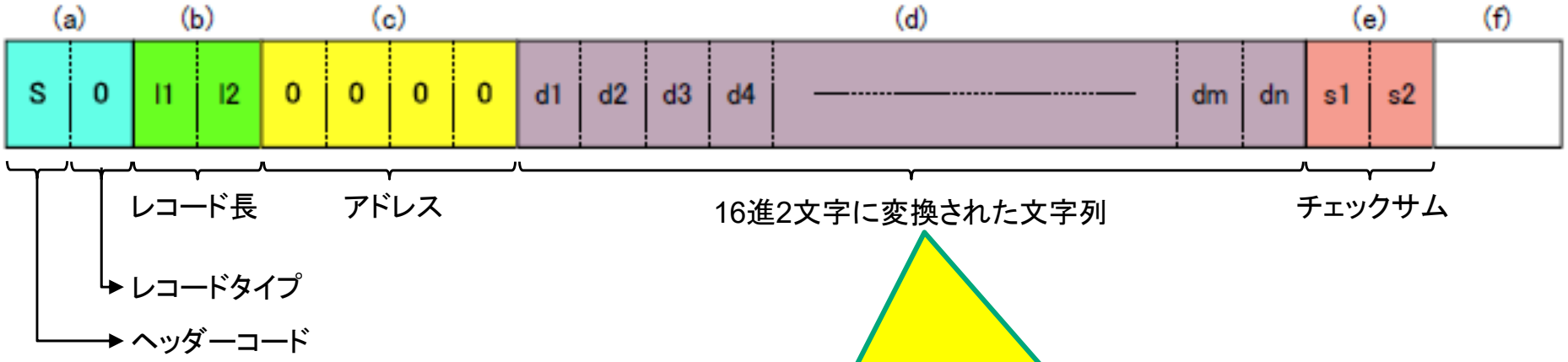
データレコードと終了レコードの違いは、データフィールドの有無

Sレコードフォーマットの詳細は、「Sレコードフォーマットの仕様.pdf」を参照のこと。

S1とS9、S2とS8、S3とS7がペアとなる。

S0のヘッダレコード

S0レコードは、ヘッダレコードで、データ部分は、ASCII文字列となっている。
文字列を含むかどうかは、任意である。

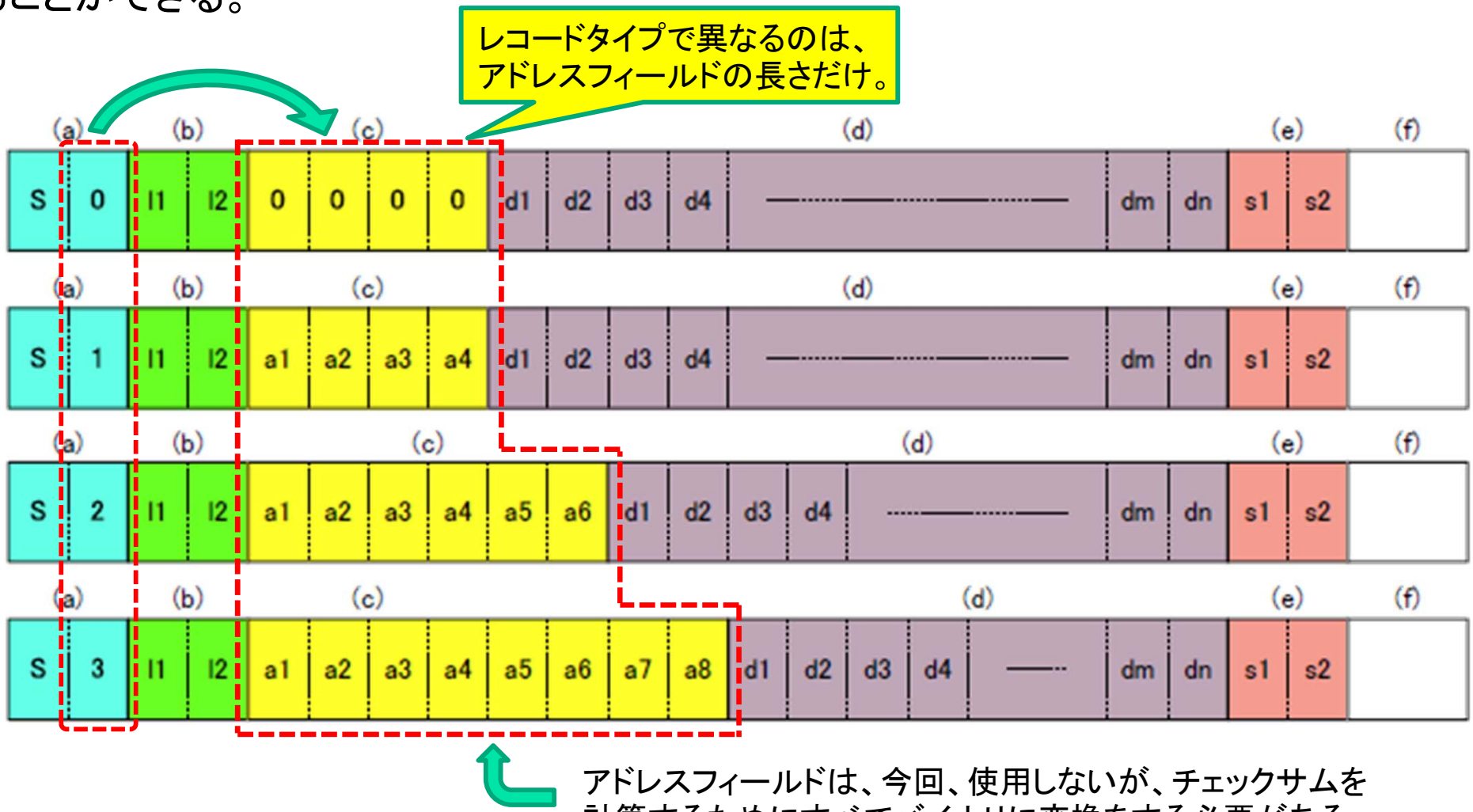


S0の他に、S5とS6タイプがあるが、今回はサポートしなくて良い。

S0レコードは例外で、データ部分は16進2文字のASCII文字列でコメントフィールドとして使用する。レコード長から、アドレス長(2バイト)とチェックサム(1バイト)を引いた数が文字数となる。その結果、0の場合は、文字列がないことを意味する

変換の考え方

Sレコード変換のアルゴリズムは、比較的簡単である。
S0～S9のタイプがあるが、違いはアドレスフィールドのバイト数だけである。
そのため、各タイプによって、アドレス長を調整すれば、以後の処理はすべて共通にすることができる。



/r (rewrite)オプションについて

- /r オプションは、誤操作で出力ファイルに上書きをさせないためのものです
  - fopen()で書き込みモードでファイルをオープンした場合、すでに、そのファイルが存在するとファイルの内容が消えてしまいます。
- fopen()の動作は、
  - モードに、”r” ”(読み込み)を指定した場合、
    - ファイルが存在すればオープンします。
    - ファイルがない場合は、エラーを返します。
  - モードに、”w”(書き込み)を指定した場合、
    - ファイルが存在しない場合は、新しくファイルを作成します。
    - ファイルが存在する場合は、新たにファイルは作成せずに、そのファイルサイズが0になります。
- 実際の処理について
  - 指定されたファイルを書き込みモードでopenする前に、読み込みモードでopenしてみる。
  - もし、読み込みモードでopenに失敗すれば、ファイルが存在していないことになる。
  - もし、読み込みモードでopenに成功すれば、ファイルが存在していることになり、
    - 開いたファイルをcloseして、
    - /r オプションが指定されていれば、fopenの書き込みモードでファイルを開く
    - /r オプションが指定されていなければ、ファイルがすでに存在しているためエラーとする
      あるいは、上書きをしても良いかと尋ねる

バイナリからSレコードにするコマンド
bin2srec (仮称)

バイナリデータからSレコード変換(bin2srec)

- bin2srecとsrec2binは、ペアで使用するコマンドです。
  - バイナリからSレコード ⇔　Sレコードからバイナリ
- バイナリデータからSレコード変換は、オプションの数が多くなります。
  - /s　レコードのタイプ(デフォルトは、1)。
  - /d　レコードのサイズ(デフォルトは、32バイト)。
  - /t　S0に入れるタイトル。
- /z オプションの活用
  - /z オプションを利用して、一度に複数のファイルの変換ができるようにします。
  - /e オプションは、/zオプションを使用した時、出力ファイルの拡張子を指定します。
- バイナリから16進文字コードの変換について
  - Sレコードからバイナリ変換は、自分で16進文字からバイナリに変換しましたが、バイナリデータからSレコードーは、printfの”%02X”で出力してかまいません。
  - もちろん、自分で関数を作成しても構いません。

バイナリからSレコードにするコマンドのシンタックス

```
C:¥Users¥m-hoshi>bin2srec /?
構文: bin2srec [<opts>] [<inpath>] [<outpath>] [<opts>]
機能: ファイルの内容をSレコードに変換
オプション:
   /r              出力ファイルが存在するとき、強制的に上書きをする
   /a[=]<hex>      ロードアドレスの指定(省略の場合は0000)
   /i[=]<file>     入力パス(<file>省略時は stdin)
   /o[=]<file>     出力パス(<file>省略時は stdout)
   /s[=]<n>        Sレコードのタイプの指定 1,2,3 (デフォルト=1)
   /d[=]<n>        1行のデータ長 16,32,64バイト(デフォルト=32)
   /t[=]<text>     S0レコードに含めるテキスト
   /z[[=]<file>]] 変換するファイル名を指定したファイルから読み込む(デフォルト=stdin)
   /e[=]<text>     zオプション時、出力ファイルに付ける拡張子(デフォルト=.txt)
   /?              ヘルプメッセージ
```

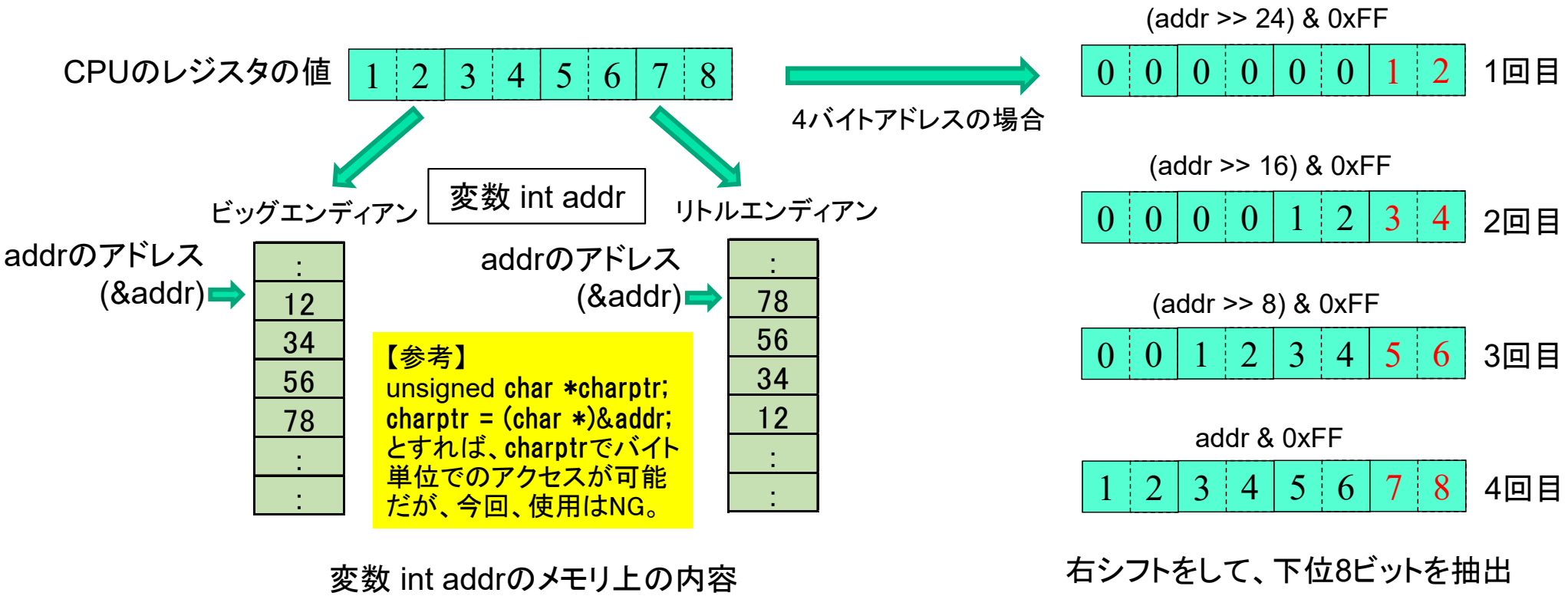
/z 、/eオプションは、処理が複雑になるため、今回はやらなくてもOKです。

バイナリからSレコード変換のヒント

- main関数でコマンド引数の解析が終わったら、switch-case文で、レコードタイプを調べて、アドレス長と、終了レコードのタイプを設定しておく
  - タイプが「1」なら終了レコードは、「9」
  - タイプが「2」なら終了レコードは、「8」
  - タイプが「3」なら終了レコードは、「7」
- そうしてから、whileループで、ファイルを読込む
  - バイナリの読み込みなので、dumpコマンドと同じfreadを使う
  - このとき、1回のfreadで読込むバイト数は、データ長で指定したサイズになる
  - そして、読込んだデータとサイズを引数にして、バイナリからSレコード変換の関数を呼ぶ
- バイナリからSレコード変換で、ちょっと面倒なのがアドレスの出力
  - 単にアドレスだけの出力であれば、”%04X”、”%06X”、”%08X”で澄む
  - しかし、アドレスも1バイトごとチェックサムに足しこむ必要がある
- データの出力は簡単で、読込んだデータサイズ分をforループで出力
  - 単純に、”%02X”で出力
- 最後は、チェックサムの出力
  - チェックサムは、値をビット反転(~)出力します。ビット反転とは1の補数にする
  - ビット反転をすると、これは、0xFFから引き算をしたと同じ値になる

アドレスの出力のヒント

- バイナリからSレコード変換で、ちょっと面倒なのがアドレスの出力
  - 単なる16進文字列の出力だけであれば、“%04X”、“%06X”、“%08X”で可能
    - しかし、チェックサムの計算があるので、1バイトずつ出力する必要がある
  - dumpのところで、ビッグエンディアンとリトルエンディアンの説明をした
    - Windows PCはリトルエンディアンなので、変数の値は、メモリ中に逆順で格納されている
  - エンディアンを決めて処理するなら、char型ポインタで1バイトずつ操作も可能だが、エンディアンに依存するプログラムを書いてはダメ。➔ シフトを使った出力方法を学んでください。
    - シフト動作は、CPUのレジスタで行われる



/z オプションの説明

- /z オプションは、変換するファイル名を/zで指定されたファイル名リストから読み込みます。
- ファイル名を与えず /z だけにすると標準入力(stdin)から読み込みます
- /zがない場合は、stdinからファイル名を入力します。

例えば、filelist.txt に下記の内容になっているとします。
fileA
fileB
fileC
  :
fileX

そのとき、コマンドラインから、
bin2srec /z= filelist.txt と入力すると、Sレコードに変換するファイル名をfilelist.txtから読み込んで処理します。

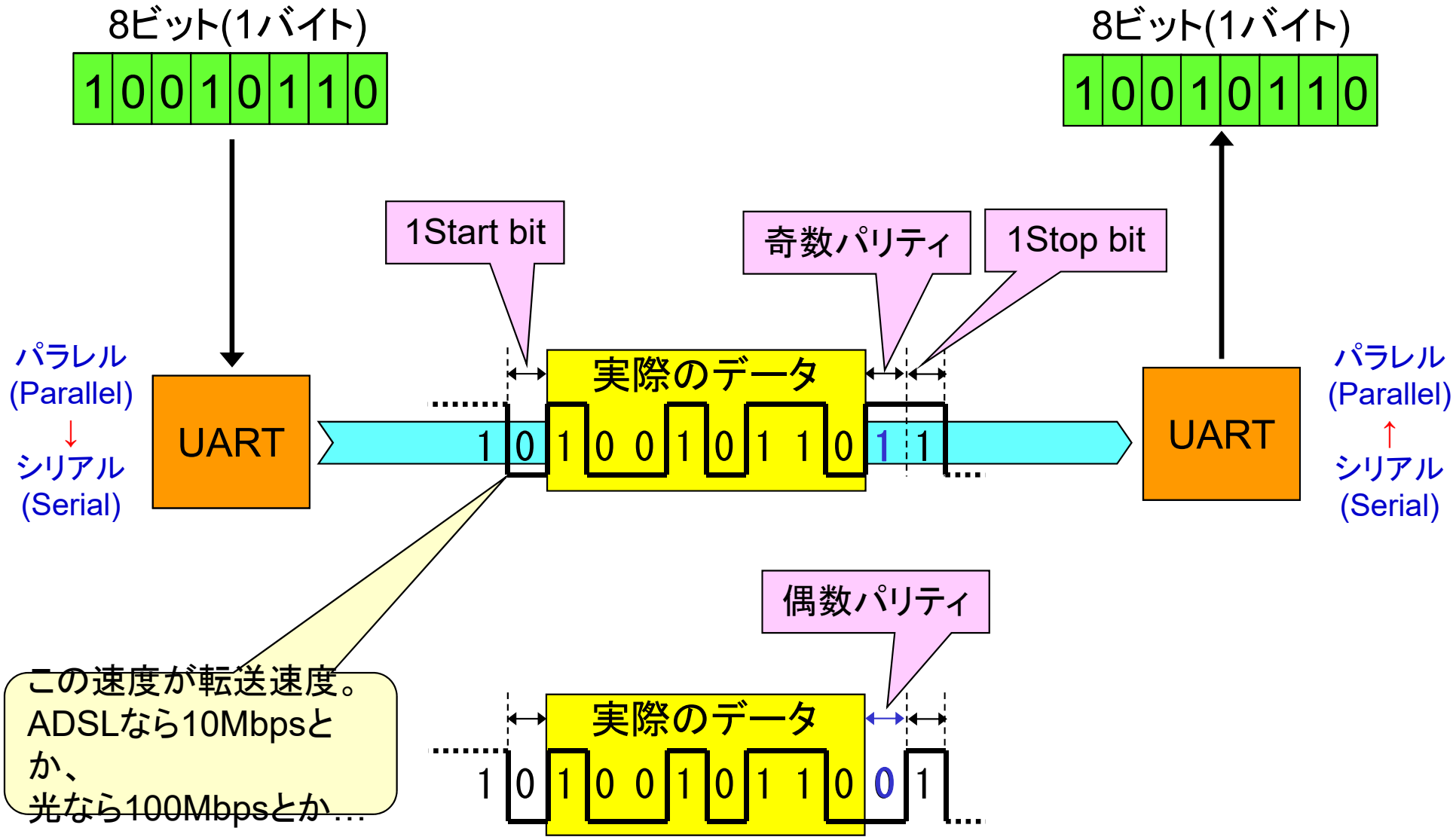
/zオプションで、ファイル名リストのファイルの指定がない場合は、stdinからファイル名を入力します。
この方法を応用すると、パイプラインを使って、例えば、

dir /b | bin2srec /z　　← dir /b の/bオプションは、ファイル名のみを出力する

のように入力すると、dirで出力されたファイルのすべてがSレコードに変換されます。

シリアル通信の参考資料

シリアル転送の仕組み



UART(Universal Asynchronous Receiver Transmitter)

非同期歩調式のシリアル通信

- RS-232C (Recommended Standard 232 version C)とは
  - シリアルポートのインターフェース規格
- UART (Universal Asynchronous Receiver Transmitter)とは
  - シリアル通信を実現する回路を表す
- RS-232CもUARTも同じ意味で使われる
  - UART: 一般的なシリアル通信全般を示す。
  - RS-232C: 電気信号などの物理的な規格を規定
- RS-232Cの規格では、信号の電圧レベルを±5V～±15Vで規定
  - 標準では、0 = +5V～+12V、1= -5V～-12V
- COMポートとは
  - RS-232Cシリアル通信デバイスのWindowsが定めた名称

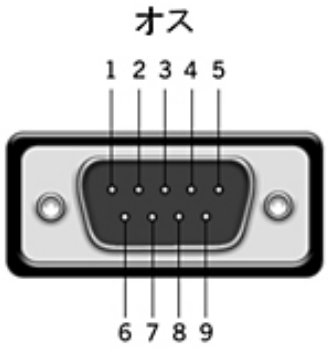
COM(RS-232C)ポートのピン配置と信号名

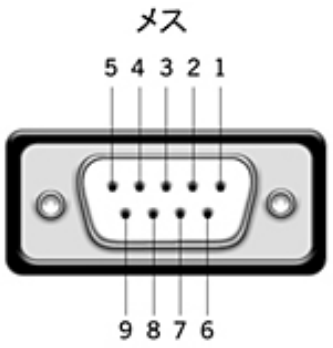
ピンーオスDTE配列の機器
（PCなど一般的な機器の配列）

ピン番号	信号名	方向	備考
1	CD	←	通常使用されない
2	RXD	←	
3	TXD	→	
4	DTR	→	
5	GND	―	
6	DSR	←	
7	RTS	→	
8	CTS	←	
9	RI	←	通常使用されない

ピンーメスDCE配列の機器
（モデムなどの配列）

ピン番号	信号名	方向	備考
1	CD	→	通常使用されない
2	RXD	→	
3	TXD	←	
4	DTR	←	
5	GND	―	
6	DSR	→	
7	RTS	←	
8	CTS	→	
9	RI	→	通常使用されない





市販されているUSBシリアル変換ケーブル

RS-232Cケーブルのクロス接続

PCとPCの1対1の接続には、クロスケーブルが必要

ピンNo.	信号名	入出力	内容
1	DCD	IN	キャリア検出
2	RxD	IN	受信データ
3	TxD	OUT	送信データ
4	DTR	OUT	データ端末レディ
5	GND	-	グランド
6	DSR	IN	データセットレディ
7	RTS	OUT	送信リクエスト
8	CTS	IN	送信可
9	RI	IN	被呼表示

PC側には、メスのコネクタが付いている