SSD Doujinshi 2

SSDの同人誌 2

TAKE FREE ¥0

待望の同人誌第2弾！基礎知識や
SSDの形状の進化をわかりやすく解説！
進化した「かんがえる自作 SSD」爆誕！

Contents

ragnag	Maker Faire Tokyo 2022 出展にあたり … 3	
とだ勝之	データセンターてんこ … 4	
ragnag	SSD とフォームファクターの進化 … 13	
Pochio	2 年ぶりの Maker Faire Tokyo と SSD 同人誌の制作 … 16	
さしすせそ	太陽が輝くとき SSD も輝く … 20	
松澤 太郎	大容量 SSD と OpenStreetMap その 2 … 24	
伊藤 晋朗	記憶するもののその先 … 28	自作 SSD 特集
余熱	かんがえる SSD … 30	
福屋 新吾	In-Storage Computation 基板「JISC-SSD」… 34	
村口	Raspberry Pi Pico SSD … 36	
じむ	漫画「ECC とは?」… 40	
川口 優樹	ECC でデータの誤りを訂正してみた … 44	
にちか	え!音が鳴る SSD でコンピューティングを!? … 48	

SSD ネタ満載の「SSD 同人誌」の PDF 版を
Maker Faire Tokyo 2021 にて公開しました。
以下のリンクから無料でダウンロードできます。
ぜひご覧ください!

https://www.kioxia.com/ja-jp/business/ssd/solution/doujinshi.html

Maker Faire Tokyo 2022 出展にあたり

ragnag

こんにちは。キオクシアです。
今年、Maker Faire Tokyo に出展いたします。実に 8 回目の出展となりました。

昨年、私たちは、キオクシアとして初めて出展し、「はたらく SSD ～ SSD の役割ってなんだろう」をテーマにキオクシアのビジネス向け SSD をご紹介しました。「自作 SSD」は大変ご好評いただき、その後もいろいろなところでご紹介する機会をいただきました。
皆さん、応援ありがとうございます。

今年のテーマは、「はたらく SSD、かんがえる SSD」です。

いま、世界を動かす大きな力は「情報」、「データ」です。
デジタルデータを支えるインフラとして、フラッシュメモリや SSD といった半導体ストレージは必要不可欠なものとなりました。スマートフォン、パソコン、そして工場の製造機器まですべてのものが IT ネットワークでつながっていますが、皆さんの利便性の高い生活や企業のデジタルトランスフォーメーションの推進には、情報・データを、蓄積するだけではなく活用する必要があります。

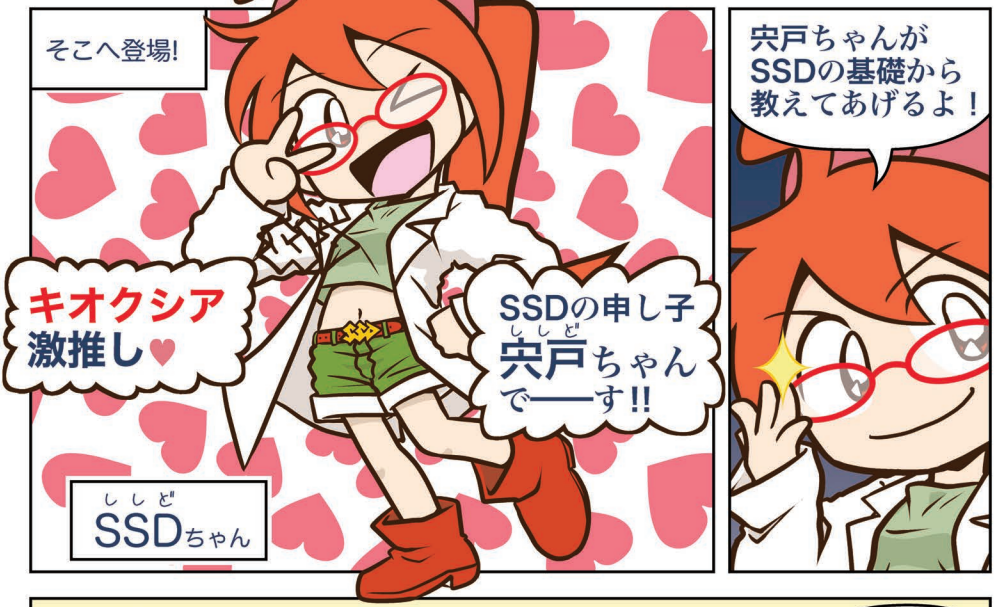
今年の自作 SSD は、蓄積したデータを活用してちょこちょこっとはたらきます。
自作 SSD も進化しているのです。

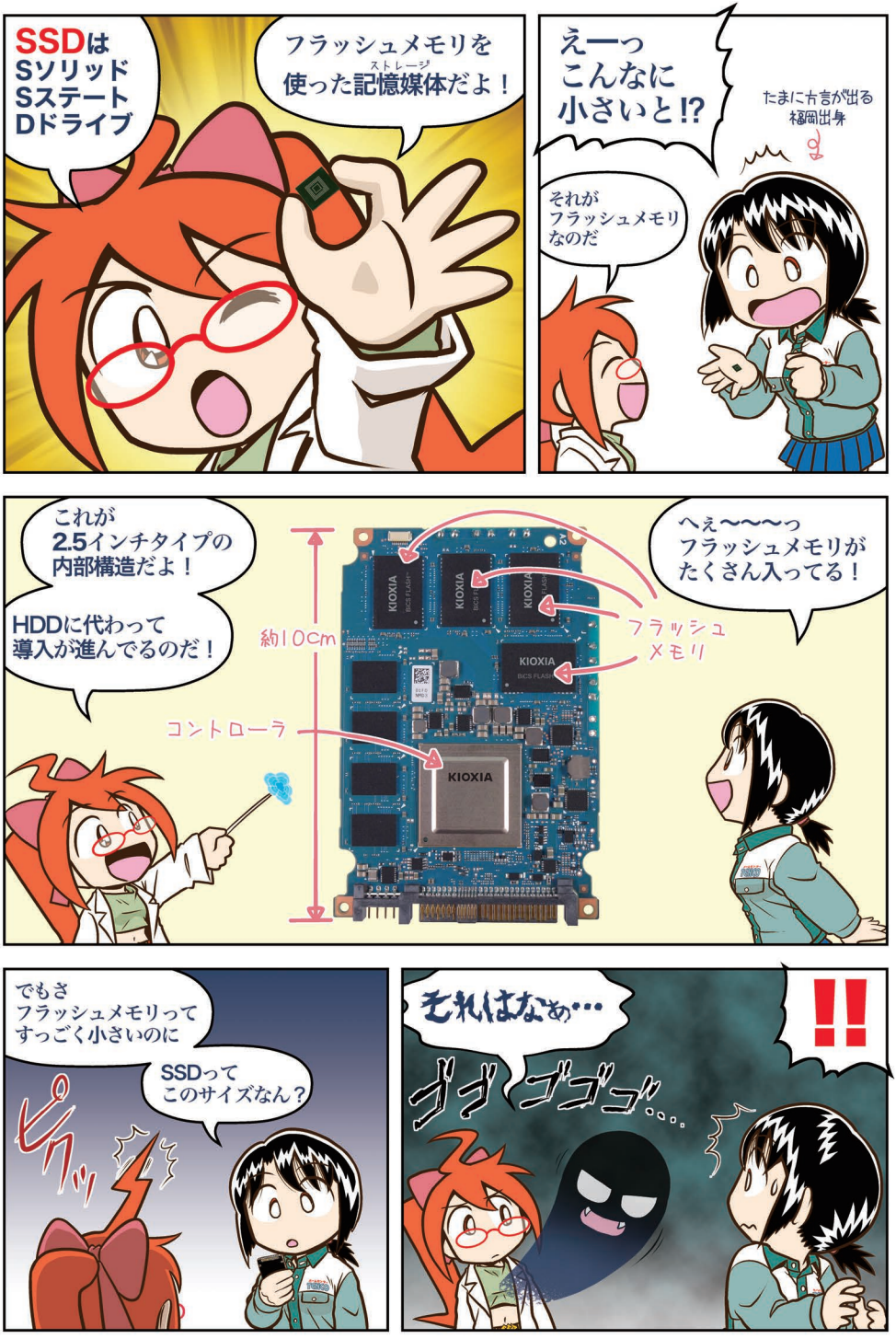
ようこそ、「はたらく SSD」の世界へ。

ragnag（@ragnag1109）
マンガを読むのと描くのをこよなく愛する広報担当。
デジタル蔵書は 500 冊を超えた！次々出てくるいろいろなフォームファクターの SSD コレクションを楽しんでいます☆
キオクシアの SSD をよろしくお願いします～！

データセンター TENCO てんこ

【前回のあらすじ】




SSDは
Sソリッド
Sステート
Dドライブ
フラッシュメモリを
使った記憶媒体(ストレージ)だよ！
えーっ
こんなに
小さいと!?
たまに方言が出る
福岡出身
それが
フラッシュメモリ
なのだ
これが
2.5インチタイプの
内部構造だよ！
HDDに代わって
導入が進んでるのだ！
約10cm
フラッシュ
メモリ
コントローラ
へぇ〜〜〜っ
フラッシュメモリが
たくさん入ってる！
でもさ
フラッシュメモリって
すっごく小さいのに
SSDって
このサイズなん？
ピクッ
それはなあ…
ゴゴゴゴゴ…
!!


HDDというぬるま湯に
すっかりなれきった者どもの
気づかぬうちに
SSDを浸透させるため
あえて同じ
フォームファクターで
作ったのだ!!
出ちゃったかぁ～
あ
フォームファクター？
外形・規格のことなのだ
うんうん
…じゃなくて
この子…誰？
ヤツらはまんまと
我らが術中にはまり
今ではSSDなしには
薄型ノートパソコンも
スマートフォンも
この世に存在できなく
なったのだぁ
もー！
言いすぎ
なのだ～
さっさと
帰るのだ～
？
？
？
あーこれは
SSDブラック
宍戸ちゃんが
思ってても言えない
心の声が思わず
外に出て来るのだ
思ってること
なんだ…
ブラックちゃん
もう少し
筐体に包んで
しゃべるのだ
わかったのだ
意外に素直…

SSDブラックの
黒豆知識

2.5インチタイプの
フォームファクター
から始まったSSDは

たとえばノートPC向けだと
より小型化した
M.2フォームファクターに
進化してゆくのだ

M.2 Type 2280

M.2 Type 2230

ちなみに2280は
幅22mm長さ80mm
という意味なのだ

M.2フォームファクターと
言っても同じ性能だと
思うなよ…

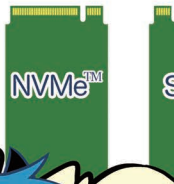
NVMe™とSATAとでは
雲泥の差なのだ！

M.2 SATAの転送速度が
約500MB/s*1に対し
M.2 NVMeのそれは
約3000MB/s*2なのだ!!

通常の3倍以上の
スピードなのだ！

それを見分けるには
端子の切り欠きを
見るのだ!!

端子が2つに
分かれているのが
NVMe!!
3つに分かれているのが
SATAなのだ!!



SSDブラックの
黒豆知識

*1)Serial ATA Revision 3.0の実効転送速度　*2)PCIe® Gen3 x4の実効転送速度

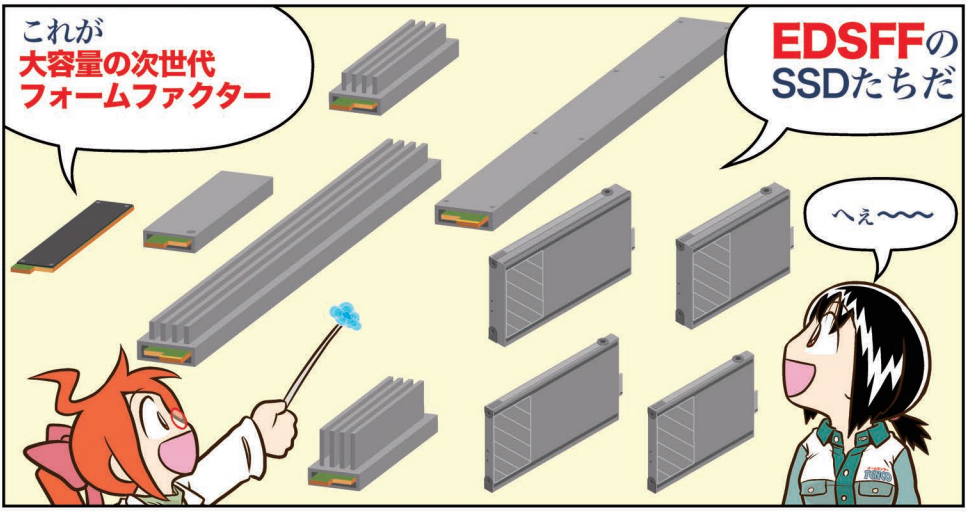
間違うとガッカリ度
3倍増しなのだ…

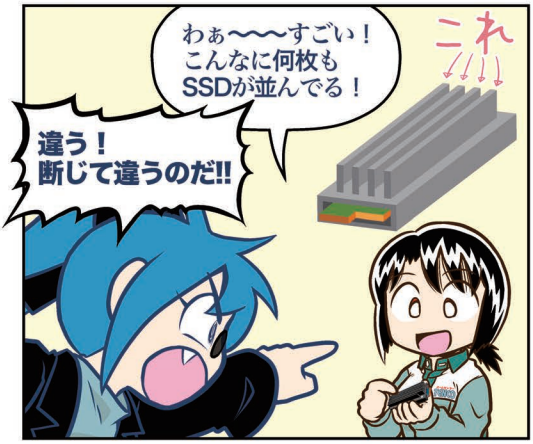
間違ったんだ…

あのときは
悲しかったのだ…

ともかく
M.2という
フォームファクターで
小型化した一方

SSDは**大容量化**
でも活躍するぞ！


これが
大容量の次世代
フォームファクター
EDSFFの
SSDたちだ
へぇ〜〜〜


わぁ〜〜〜すごい！
こんなに何枚も
SSDが並んでる！
これ
違う！
断じて違うのだ!!


これは放熱のための
冷却フィン（ヒートシンク）なのだ


おまえ今
「進化したのに熱くなるの〜？」
とか思っただろ！
だがな！
SSDの大容量化と
高速化によって
発熱量が多くなるのは
仕方ないことなのだ…
宿命なのだ！


わかるよ!!
ブラックちゃん！
ドキッ
…へ？


わたしも
数学やっとると
熱が出るもん！
ぷしゅ～
わ…わかれば
いいのだ…
その熱を
冷ますための
フィンなのだ
ホントだ
触るとなんか
冷たいね
15mm
もっとフィンが
デカいのもあるぞ！
ドキ
ドキ
でもせっかく
小さく出来るのに
これじゃノーパソに
入れられないね
ち…違うのだ
それはノーパソ用
ではないのだ
これから世界で扱う
データ量はどんどん
増えていき
いずれは
ゼタバイト(ZB)
ヨタバイト(YB)
の時代が来るのだ

TBはわかるけど…
酔ったバイト
お寺でバイト
あれ？神社？
あんちゃん運転荒えなぁ
ぜったいよった
うぷ
宅配
ゼタ？ヨタ？
SSDブラックの黒豆知識
1000バイト＝1kB(キロバイト)
1000kB=1MB(メガバイト)
1000MB=1GB(ギガバイト)
1000GB=1TB(テラバイト)
1000TB＝1PB(ペタバイト)
1000PB=1EB(エクサバイト)
1000EB=1ZB(ゼタバイト)
1000ZB=1YB(ヨタバイト)
1ZBは10億TB
1YBはその1000倍なのだ
銀河系の星の数が10ZBと言われておるから
1YBは銀河系100個ってことだな
ふぇ〜〜
なんかブラックちゃん優しくなったね！
おねえちゃんが心を癒やしてくれたからなのだ
そ…そんなことないのだ!!
そんなこと言うとこの318.75mmSSD E1.Lが炸裂するだぞ!!
長っ!!
も…もう帰るのだ！
いろいろ教えてくれてありがとー！
またねー
おぼえてろよ！
照れてるのだ


でも…そこまで
大容量だと
この大きさに
収めるのって
無理じゃない？
今はな
そのかわり
4TBのEDSFF SSDを
32個入れた
サーバーがある
サーバー
そのサーバーを
8台入れた
ラックがある
SSD
4×32=128
128×8=
えっと…
サーバーラック
そのラックを
大量に並べた
データセンター
が既にあるのだ
ああ！
こういうのテレビで
見たことある!!
つまりこのラックを
どんどん増やせば
ZBもYBも
夢じゃないって
ことやね！
SSDすご――い!!

宍戸ちゃんのパパとは一体？
果たして2人はデータセンターを見られるのか!?

つづく

SSD とフォームファクターの進化

ragnag

近年、パソコンの内蔵ストレージとして SSD の普及が進み、皆さんが購入するメーカー製パソコンの多くに SSD が内蔵されるようになりました。SSD は、2000 年代から、HDD(ハードディスク) に代わるストレージとして登場し、一般向けには、主に HDD からの交換用として HDD と同形状の 2.5 インチと、チューイングガム形状に似た M.2（エム・ドット・ツー）という基板タイプの製品が販売されています。

ビジネス向け SSD も、2000 年代から始まったインターネット・クラウドの普及に伴い、モバイル PC 製品向けの SSD からサーバー・データセンター向け SSD へと市場が拡大し、SSD のフォームファクターに進化が見られます。

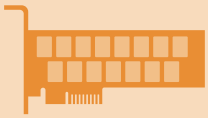
EDSFF（**E**nterprise and **D**atacenter **S**tandard **F**orm **F**actor）は、次世代のハイパースケールデータセンターやエンタープライズデータセンター向けのフォームファクターです。標準化団体により策定された基本仕様に準拠した SSD を多くのメーカーが開発を進めています。

SSD フォームファクターの進化

EDSFF が登場する前は、NVMe™ SSD のフォームファクターは 2.5 インチ (U.2)[1] と M.2 が主流でした。2.5 インチ (U.2) は、HDD と同じ形状で、多くのノート PC やデスクトップ PC に使用されています。2.5 インチ (U.2) の SSD は、通常、大容量ストレージが必要なハイエンドのワークステーション、サーバー、エンタープライズアプリケーションで使用されています。M.2 より高い動作温度に対応しています。

M.2 は、主に薄型ノート PC に内蔵することを目的としたフォームファクターで、マザーボードに直接取り付けます。M.2 には長さが異なるフォームファクターがあり、長い形状の SSD は、多くの NAND フラッシュメモリチップを取り付けることができるため、大容量になる傾向があります。

1　NVMe™ だけでなく SCSI ベースの SAS や SATA で接続するものを 2.5 インチ、少し厚めで異なるコネクタを使用し、PCIe® で接続するものを U.2 と分類することもあります。



Add-in Card (AIC)

High Performance Storage
Server Accelerator

2.5-inch (U.2)

Data Storage
Cache
Client, Servers, Storage

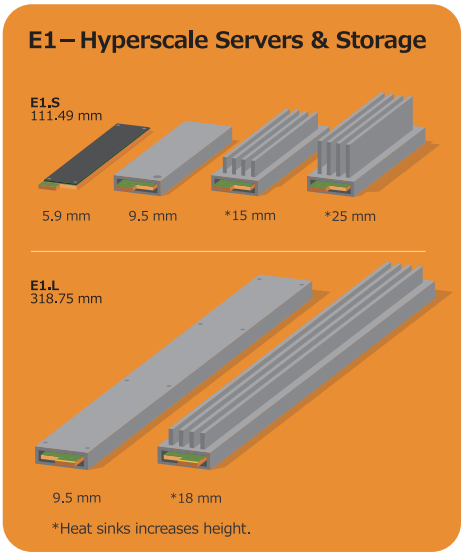
M.2
(2242, 2280, 22110)
Data Storage
Boot
Client, Servers

BGA (16x20mm)
M.2 (2230)
Data Storage
Boot
Laptop, Tablet

キオクシア事例

次世代フォームファクター EDSFF

EDSFF は、下図のとおり多くのフォームファクターがあります。インターフェイスは PCIe® 4.0 や PCIe 5.0 などに対応しています。



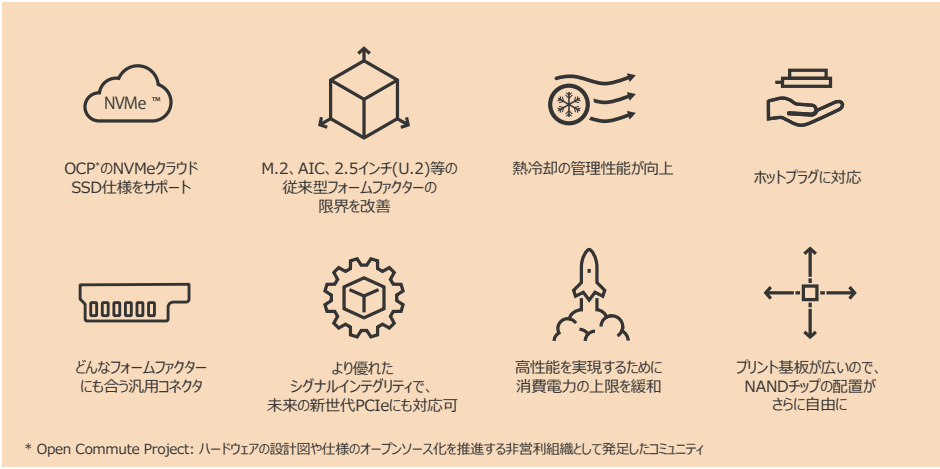
ハイパースケールサーバー、ストレージ向けの E1 ファミリーには、「E1.S」と「E1.L」があります。「S」は「Short」、「L」は「Long」の略で、SSD の筐体の長さを示しており、

E1.L は見た目が定規（Ruler）に似ているため、「Ruler」スタイルとも呼ばれます。E1.S は M.2 とほぼ同じ長さで、ハイパースケールデータセンターで使われている M.2 フォームファクターの後継になると考えられています。横方向に並べて配置できるため、筐体の幅に合わせてさまざまな容量に対応できます。また、アクセス速度が速いので、データを高速処理する時に消費電力が増し、高温になる場合があるため、冷却機能を備えています。

エンタープライズサーバー、ストレージ向けの E3 ファミリーは、U.2 フォームファクターの後継となっており、こちらも長短のバリエーションを持っています。

EDSFF SSD 採用によるメリット

EDSFF SSD 採用によるメリットとしては以下が挙げられます。



EDSFF を、もっと詳しく知りたい方は、下記も参考にしてください。

- KIOXIA ビジネス向け SSD ホームページ
- EDSFF(Enterprise and Datacenter Standard Form Factor)（英語）

https://business.kioxia.com/content/dam/kioxia/shared/business/ssd/doc/EDSFF_Infographic.pdf

2 年ぶりの Maker Faire Tokyo と SSD 同人誌の制作

Pochio

またしても同人誌制作の夏がやってきました。お手にとっていただきましたこの SSD 同人誌第 2 号は、Maker Faire Tokyo の出展に合わせて制作する通算 8 冊目の本になります。昨年同様コロナ禍のため、在宅勤務をしつつ編集に取り組んでいます。

さて、いつものように昨年の出展についてまとめたいと思います。昨年は開催の 1 か月前にコロナ感染者数が急増し、急遽オンライン開催となりました。これに伴い、はじめての動画制作に挑戦いたしました。この顛末を簡単ですがご紹介しようと思います。

気分新たに SSD 同人誌を制作

当社の Maker Faire Tokyo へのスポンサー出展は、2019 年を最後に途絶えておりました。ところが昨年は思いもよらないきっかけで、2 年ぶりに参加できることになりました。キオクシアに社名が変わってから初めての参加です。スポンサーに復帰できるとなれば、復活のご要望がとても多かった同人誌を再び・・・と考えました。出展のテーマが法人向け（いわゆる B2B 向け）の SSD 製品だったため、記念すべき新しいタイトルは「SSD 同人誌」となりました。

図 1: SSD 同人誌第 1 号の表紙

この同人誌では、初の試みにいくつか取り組んでいます。まず、キオクシア初の公式同人誌となっています。つまり社内でいろいろとお許しを得るのに頑張ったということです（汗）。それから SSD にまつわるトピックをとだ勝之先生のまんがで紹介する企画を立案し、その実現に動き出しました。とだ先生には、弊社が製作に協力し発刊しました「学研まんがでよくわかるシリーズ 144 フラッシュメモリのひみつ」で大変お世話になりましたので、機会があればぜひ新作をお願いしたいと考えていたのでした。

新作を描いていただくにあたり、新しいキャラクターや世界観をゼロから起こすのはとても大変な作業で、残された制作期間を鑑みるとかなり難しいことがわかりました。そこで、とだ先生がかつて月刊誌で連載されていた「ホームセンターてんこ」の世界で SSD を紹介する、「データセンターてんこ」を描いていただくことになりました。また、折角です

のでこのまんがのために新キャラクターを作っていただきました。その名も SSD にとても詳しい「宍戸ちゃん」です。ところがまんがのストーリーを検討する時間が少なかったので、SSD 同人誌第 1 号では 2 ページの描き下ろしまんがとなりました。「続きは WEB で?」と思わせぶりな終わり方になったのに、WEB での続編発表ができないままでしたが、幸いこの SSD 同人誌第 2 号に続編を掲載できることになりましたので、ぜひお読みいただけますと幸いです。

また、とだ先生には同人誌の表紙イラストも描き下ろしていただきました（図 1）。とだ先生のイラストだと気が付いて本を手に取ってくださった方がいらっしゃると聞きましたので、この新刊もとだ先生のファンの方のご期待に沿うことができれば幸いです。

自作 SSD ネタが話題に

SSD 同人誌第 1 号では SSD の自作に関する記事をメイントピックとして掲載しました。この自作 SSD ネタは執筆陣が企画前から温めていたネタで、このたび満を持して実行に移したものです。素のフラッシュメモリのチップ、いわゆる Raw NAND を使いますが、単体のチップだけでは普通に使えません。そこでマイコンボードと自作プログラムにより、SSD のコントローラに相当するものを構築します。使用したフラッシュメモリのチップの容量は 1Gbit でしたが、実際に自作 SSD として使用できる容量は約 2MByte（1Byte=8bit なので、約 16Mbit）でした。今年はどこまで進化したのか、ぜひ本誌の関連記事をご覧ください。

また、この自作 SSD を自作 CPU と組み合わせて活用することで、その自作 CPU に「生命、宇宙、そして万物についての究極の疑問の答え」を計算させるという、壮大に見えるネタ的な応用事例も掲載しました。さらに同人誌の後半では、SSD コントローラについて専門家によるやさしい解説記事や、法人向けの大容量 SSD の読み書きスピードを OpenStreetMap を使って実験、検証した記事、そしてシングルボードコンピュータに NVMe™ SSD を搭載しベンチマーク実験を行った記事も掲載しました。FlashAir™ 同人誌のころからの読者はお気づきだったかもしれませんが、恒例のソラちゃんまんがの新作も掲載しました。

延べ約 1 万人の読者

ところでこの SSD 同人誌第 1 号ですが、先に述べた通りオンライン開催となったため会場での冊子の配布が不可能となりました。そこで例年では Maker Faire Tokyo の開催後しばらくしてから公開している PDF 版を、オライリー・ジャパン様のご協力によりオンライン開催日にあわせて当社の出展内容紹介ページにて公開させていただきました。こちらは “SSD 同人誌 ” で検索いただければ、すぐにお読みいただけます。

ありがたいことに PDF のダウンロード件数は 6600 を超え、大変好評でした。また、動画を視聴してアンケートに回答いただいた方に、抽選で冊子版をお送りさせていただきました。こちらもオライリー・ジャパン様にお力添えいただきました。この場を借りて厚く御礼申し上げます。

その冊子版ですが、その後合計で約 3500 部を各所にて配布させていただきました。したがって冊子版と PDF 版を合わせると、延べ 1 万人近くの方に同人誌をご覧いただいたことになります。お読みいただいた皆様、ありがとうございました。

ヨネミナー動画の制作と公開

突然の動画制作

開催約 1 か月前に Maker Faire Tokyo 2021 がオンライン開催となったため、急遽動画を制作することになりました。動画の内容は同人誌の記事と同じく、「法人向け SSD のご紹介」と「自作 SSD」の 2 本立てにすることにしました。とはいえ、これらを真面目にプレゼンしたら Maker Faire っぽさがなくなってしまうと感じました。年に一度の楽しいイベントですから、動画も Maker の方たちに喜んでもらえるものにしたいなと。

そこで、当社は様々な展示会に出展することがあるのですが、その一部のイベントで好評いただいている余熱によるセミナー、通称「ヨネミナー」をオンラインで開催することにしました。ところがこれが中々大変でして、当然ですが権利的に問題がない動画を制作しなくてはなりません。実はヨネミナーでは通常、みなさんよくご存じのフリーで使用可能なほんわかテイストのイラスト集を素材に使用するのですが、権利関係を鑑み今回はイラストを自作することにしました。作画はこれまで同人誌をご愛読いただいた方にはおなじみの、ソラちゃんの生みの親にお願いしました。

あつまれ！はたらく SSD

1 本目の動画は「あつまれ！はたらく SSD」と題した、法人向け SSD のご紹介です（図 2）。年齢不詳の「たっくん」が、寝る前にお父さんに「はたらく SSD」という絵本を読んでもらうという、お子様のいらっしゃるご家庭でよくあるシチュエーションです（図 3）。ところがたっくんがパパよりやたら SSD に詳しいのです。これ以上紹介するとネタバレになってしまいますので、興味をお持ちの方はぜひ動画本編をご覧ください。

ところで 1 本目と 2 本目の動画の間に余熱本人が登場して、「キオクシアってなんだろう、レトロゲームかな?」と話しているのですが、何のことかわからない方が多かったのではと思います。実は権利の関係で元ネタがわかる絵を載せられなくなってし

図 2: あつまれ！はたらく SSD

図 3: たっくんとパパ

まって差し替えたので（図 4）、ある程度の年齢層の方でないと元ネタに気づけなかったかもしれません。気になった方はさほどいないと思いますが、一応ヒントを載せておきますと、国内の有名パソコンゲームメーカーが 1986 年に発売した、ロールプレイングゲームシリーズの 3 作目です。

図 4: 大人の事情で画像を差し替え

はじめての SSD 自作

2 本目の動画は自作 SSD ネタをテーマにした「はじめての SSD 自作」（図 5）です。こちら、「おーい磯野！メモリコントローラ自作しようぜ！」（図 6）というパワーワードで始まりますが、どこか懐かしさを感じずにはいられません（汗）。作品 No. がなぜ 1049 なのか分かった方はすごいですよ。前半の寸劇？のあとは、余熱による自作 SSD の解説があります。こちらもこれ以上ご紹介しますとネタバレになってしまうので、ぜひ動画本編をご覧いただけますと幸いです。

この 2 本立ての動画は「あつまれ！はたらく SSD」で検索するとご覧いただけます。全長 20 分ですので、よろしければぜひご覧ください。また SNS などでこの動画だけでなく本同人誌の感想もお寄せいただけますと大変ありがたいです。ハッシュタグ「#SSD 同人誌」でよろしくお願いいたします。

図 5: はじめての SSD 自作

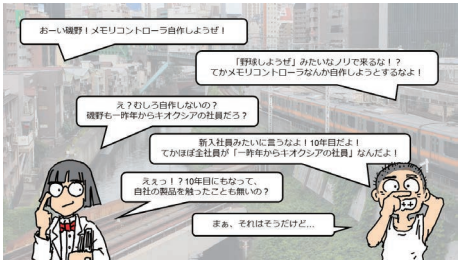


図 6: 磯野と中島

Pochio (@I_love_nintendo)

実は余熱が締め切り 1 日前に自作 SSD を某誌のラズパイコンテストに応募したら、なんと優秀賞を頂いてしまいました。ありがとうございました。来年は自作 SSD の基板を使った様々な製作記事で同人誌を作りたいです。ぜひみなさんチャレンジしてみてください。

太陽が輝くとき SSD も輝く

さしすせそ

何言ってんの?って感じですが、あながち間違いではありません。SSD で使われている記憶素子 “ フラッシュメモリ ” が情報を記憶するしくみと、太陽が輝くしくみには共通点があり、どちらも量子論の “ トンネル効果 ” がカギとなっています。今回はその紹介です。

トンネル効果： ミクロな粒子が壁をすり抜ける

原子や電子などのふるまいを表す量子論で有名な現象に “ トンネル効果 ” があります。通り抜けられないはずの壁をミクロな粒子が “ ある確率で ” すり抜けてしまう現象のことです (図1)。ここでいう壁とは、例えば、電子にとっては絶縁体のことで、電流が流れない領域は壁に見えます。もう一つの例としては、プラスの電気を帯びた粒子は、別のプラス粒子の近傍で反発力を受けるため、壁があるように見えます。ミクロな粒子が、それらの壁をある確率ですり抜けてしまう現象を、トンネル効果と呼んでいます。

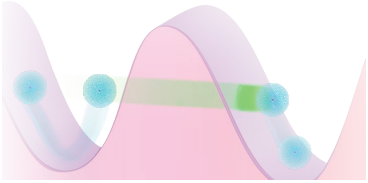


図 1: トンネル効果のイメージ

太陽の輝き： トンネル効果が不可欠

太陽では、水素からヘリウムを生成する核融合反応が起きています。その際にエネルギーを放出し “ 輝いて ” います。核融合が起きるには、プラスを帯びた原子核どうしが電気的な反発力 (壁) を越えて近づく必要があります。もし、粒子が充分に高速で衝突すれば、反発力の壁を乗り越えることができます。しかし、そのためには数百億度に相当する粒子の運動エネルギーが必要です。太陽の実際の温度は高々 1500 万度程度なので、このままでは反発力の壁を乗り越えられず、太陽は輝けません。そこでトンネル効果の登場です。トンネル効果があると、電気的な反発力の壁をすり抜ける原子核が現れてきます。一旦壁を抜けると、今度は原子核付近のみに働く “ 強い力 ”(引力) の作用で、核融合反応が進みエネルギーが放出されます。つまり、太陽はトンネル効果のおかげで輝いているのです。

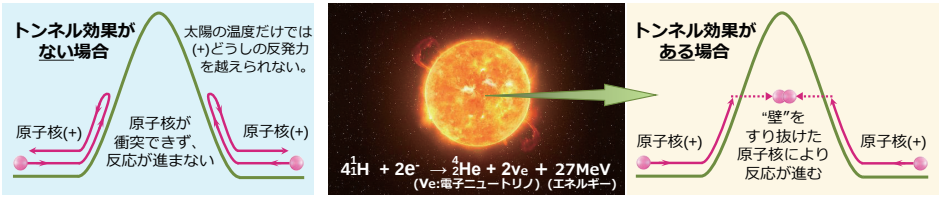


図 2: 太陽の核融合反応におけるトンネル効果の役割

SSD： 半導体のフラッシュメモリに情報を記憶。フラッシュする（輝く）メモリ？

SSD などの記憶素子として用いられるフラッシュメモリは、写真や文字などの情報を記憶できる電子部品です。記憶した情報をカメラのフラッシュ（光）のように“パッ”と一度に消すことができるので、そう名付けられました。ここでは、半導体であるフラッシュメモリの動作原理、特にトンネル効果を使った記憶のしくみについて、順番に説明していきます。

図 3: フラッシュ（輝き）メモリ

半導体： 抵抗率が大きく変わる性質を利用、MOS トランジスタが代表例

半導体とは、もともとは物質の電気的性質を表す言葉です。物質を抵抗率で分けると、電気を通す導体と、通さない絶縁体があり、その中間に半導体があります（図4）。半導体には面白い性質を持つものがあり、光や電界などを与えると抵抗率が大きく変わるモノがあります。微小なスイッチや増幅器として活用できるために広く使われています。代表はMOSトランジスタです。

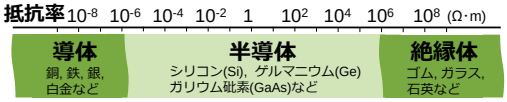


図 4: 半導体：抵抗率による分類

MOS トランジスタ： 目に見えないほど小さなスイッチ。増幅作用も。

トランジスタとは電流を制御する部品のことです。MOS の名は、その積層構造（図5）に由来します。上から Metal（導体）, Oxide（酸化膜：絶縁体）, Semiconductor（半導体）となっています。3 つの電極があり、ソース・ドレイン・ゲートと呼ばれます。ゲート電圧 Vg が、しきい値電圧 Vth を越えると、ソース・ドレイン間に電流 Ids が流れるため、スイッチとして働きます（図6）。電流が流れる領域は、チャネルと呼ばれ（図5）、ゲート直下の半導体表面に形成され、深さは 10nm（ナノメートル）程度です。スイッチ動作するトランジスタを組み合わせると、コンピュータなどに使われる論理演算回路を作成できます。別の動作方法として、ゲート電圧を図6の (+) 値付近で小刻みに動かすと、対応する電流 Ids が大きく振れる増幅作用が得られ、アンプやアナログ回路の構成要素になります。

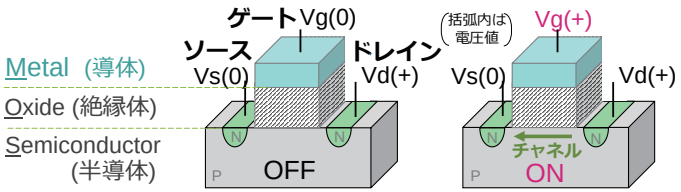


図 5: MOS トランジスタの構造

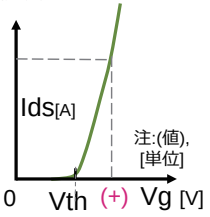


図 6: 電気特性

MOS トランジスタは、４０年以上の継続的な微細化により、既に光学顕微鏡では見えないほど微細化が進んでいます。最近では、数十億個の MOS トランジスタを集積したチップが製品化され、様々な機能を実現しています。(図9に半導体チップの例)

半導体メモリ：　MOS トランジスタベースの記憶素子。フラッシュメモリが代表例

半導体を使うと、演算だけでなく、情報を記憶することもできます。フラッシュメモリを例に、そのしくみを解説します。図7にフラッシュメモリの記憶素子の構造を示します。MOS トランジスタに電荷保持層が追加されていますが、基本動作は変わりません。一般に、情報 1bit を記憶するには、記憶素子に異なる 2 状態が必要です。フラッシュメモリの場合、情報 “0” を記憶する場合のみ電荷保持層に電子を注入します。情報の読出しは、記憶素子のトランジスタ挙動の差として検知します。具体的には、記憶素子のゲートに+電圧 Vg をかけたときに、記憶素子がスイッチ ON となれば情報 1、OFF のままなら情報 0 が記憶されていると判別します。情報 0 のときに、スイッチが OFF のままとなる理由は、ゲートに+電圧 Vg をかけても、保持層の電子 (−) が打ち消してしまい、半導体表面に充分なチャネルを形成できないためです。機器の電源を切っても、保持層の電子は周囲の酸化膜を越えられず、情報が保持される不揮発性メモリとなります。

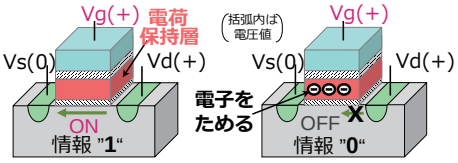


図 7: フラッシュメモリの記憶素子 (メモリセル) 構造

フラッシュメモリ：　情報の記憶にトンネル効果

フラッシュメモリの情報の記憶と消去には、絶縁体である酸化膜の壁を越えて電荷保持層へ電子を出し入れする必要があります。そこにはトンネル効果が使われています。フラッシュメモリの場合は、酸化膜に高電界を印加する FN トンネル電流 (図8) を利用しています。一例としては、酸化膜に 10MV(メガ・ボルト)/cm の高電界をマイクロ秒程度印加することで、数百個の電子を移動させることができます。

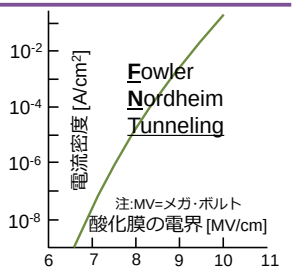


図 8: FN トンネル電流

半導体は身近な存在：　日常生活や情報化社会の発展を支える

半導体製品の多くは、MOS トランジスタやメモリセルを集積した “ チップ ”(図9) の形で製品化され、現在では電池や電源を使うほとんどの機器に使われています。2021 年頃には、半導体不足により給湯器や自動車の出荷が滞った、という報道もありました。半導体が便利な生活を支えている傍証にも聞こえます。

一方で、半導体は、発展する情報化社会の基盤も支えています。計算機の理論によると、どんな機能でもアルゴリズムで書ければ、状態遷移に従い、自動で機械的に実行することが可能です。これはハードウェア (＋ソフトウェア) で実装可能で、ハードウェアなら半導体の記憶素子と MOS トランジスタの両方を用いて “ 有限状態機械 ” を作ることで信頼性高く実装できます。例えば、皆さんのスマホなどは、この考え方に則り、様々な役割を持った半導体チップが多数搭載されていて、各種ソフトウェア (アプリ等) がハードウェア (半導体) 上で実行されています。今後も、人工知能 (AI) の発展も手伝い、今までにない機能がどんどん実装され、情報化社会がますます発展し、半導体の重要性も増していくように思います。

図 9: 半導体チップ (左) とパッケージ (右)

おわりに

太陽の輝きも、SSD の記憶素子に使われるフラッシュ (輝き) メモリも、どちらもトンネル効果で “ 輝く ” 共通点がある、というお話でした。フラッシュメモリは電源を切っても情報が消えない不揮発の特性をもち、最近では約 1cm 角のチップに 1T(テラ : 1 兆 =10^{12})bit の情報を記憶できます。DVD(4.7GB) なら 27 枚分、スマホのストレージが 128GB なら同じ記憶容量です。日常生活に不可欠でありながら、発展する情報化社会の基盤でもあります。もっと知りたい方には、無料公開の学研まんが “ フラッシュメモリのひみつ ”[1] もお勧めです。

キオクシアでは、フラッシュメモリの発明[2] と、３次元フラッシュメモリを開発[3] した技術力を基盤として、更なる研究・技術開発を推進しています。キオクシアの工場では、世界中の約 1/3 のフラッシュメモリが生産[4] され、高品質の維持のために、毎日 20 億件以上の製造・検査データが収集され、ＡＩ技術を駆使した解析が行われています。今後も、高速・大容量・高信頼性のメモリや SSD を世界中に提供していきます。

1　学研まんが “ フラッシュメモリのひみつ ”https://bpub.jp/bookbeyond/item/000405917196
2　1987 年、世界初 NAND 型フラッシュメモリを発明。
3　2007 年、世界初 3 次元フラッシュメモリ技術を発表。
4　2020 年キオクシア調べ。Western Digital 社との共同設備投資分を含む。

さしすせそ

半導体の研究開発に携わり 30 余年。これまで DRAM, CPU, BiCS FLASH™ の設計開発なども担当しました。製品チップは、皆さんのゲーム機やスマホ・SSD に入っているかも。

大容量 SSD と OpenStreetMap その 2

一般社団法人オープンストリートマップファウンデーションジャパン　松澤 太郎

昨年の同人誌では「大容量 SSD と OpenStreetMap」というタイトルで寄稿をさせていただきました。

今回、最新フォームファクター EDSFF(Enterprise and Datacenter Standard Form Factor) に準拠した最新の PCIe®4.0•NVMe™ 1.3c 対応の SSD、KIOXIA XD6 Series（図 1）をキオクシア様からお借りすることができたので、新規のベンチマーク及び新しい試みをいくつか行いました。OpenStreetMap（以下、OSM）の基本的な説明は昨年の同人誌をご覧いただきたいのですが、簡単に説明すると「世界単位の単一の地図データベース」となります。データベースの形式は XML ファイルで、世界単位のデータ[1]で 2022 年 7 月現在、bzip2 圧縮で 118GB、Protocol Buffer 形式で 65GB ほどになります。また、今回は全て世界単位のインポートについて紹介します。

図 1: KIOXIA XD6 Series

検証マシンの紹介

今回は前回実験に用いた検証マシンの ATX マザーボードから入れ替えを行いました。

- ASRock Z690 Pro RS
- Intel® Core™ i5-12400
- 16GB Memory
- SSD KXD6CRJJ3T84 (3.84TB PCIe)

大きな違いは CPU が 8core 8thread から 6core 12thread になった事と、SSD の接続が NVMe 接続から PCIe に直接挿す形になった事です。KIOXIA XD6 Series の詳細はキオクシアの HP[2] を参考にしてください。

また、前回同様 OS は Debian bullseye を利用し、PostgreSQL については pgtune[3] を元にチューニングをしています。プログラムの速度の計測については time コマンドを、ストレージのスピードの計測については Prometheus™ の node-exporter を使います。

OverPass API のデータベース作成

昨年の同人誌でもご紹介した Overpass API に使えるデータベース (独自形式) を作成するベンチマークを取得してみました。

1 https://planet.openstreetmap.org/
2 https://business.kioxia.com/ja-jp/ssd/data-center-ssd/xd6.html
3 https://pgtune.leopard.in.ua/

結果としては以下の通りとなりました。

```
time  bin/init_osm3s.sh planet-220711.osm.bz2 "db/" "." --meta
real    2036m30.440s
```

実測としては昨年のベンチマークが 36 時間に対して、今回は 34 時間と大きな差はでませんでした。

ただし、昨年から 1 年かけてデータ自体は bzip2 形式で 7GB 程度大きくなっているので、それを加味すると少なくともストレージのスピードの影響を受けていると思われます。

Imposm3 による Planet のインポート

次にこちらも昨年の同人誌でご紹介した imposm3[4] による Planet のインポートの計測をしてみました。

まずデータベースを作成します。

```
createdb -O osm imposm3_planet
psql -U osm imposm3_planet
create extension postgis;
create extension hstore;
```

では、インポートを行います。

```
time ./imposm import --connection postgis://osm:osm@localhost/imposm3_planet
-mapping mapping.json -read ../../data/planet-20220711.osm.pbf -write
[2022-07-25T03:58:44+09:00] 6:50:20 [step] Finished: Imposm in
6h50m20.956567946s
```

6 時間 50 分ほどでインポートすることができました。

昨年のベンチマークでは 10 時間 30 分ほどだったので、だいぶスピードが速くなっています。

実際のところ、どれぐらい速いのか Prometheus で計測されたグラフを見てみましょう。図 2 を参照していただきたいのですが、読み込みの実測で最大 33GB/s 出ています。

このストレージ、カタログの数字が 6500MB/s なんですが…

図 2: imposm3 のストレージ (read)

4 https://imposm.org/

このプロセスは一旦キャッシュをストレージ上に作成してから処理を行うため、だいぶこのキャッシュアルゴリズムが優秀で、かつインポートプロセスがマルチプロセスであるということでこのようなスペックが出ているのかと思います。というか PCIe 直付け恐ろしいですね…

書き込みの実測でも瞬間的に 5GB/s ほどの数字が出ているので、昨年検証した KIOXIA XD5 Series に比べて非常に速い事がわかります。

Valhalla による世界単位での経路探索

昨年の同人誌では pgRouting というソフトウェアを使いましたが、今回は世界単位でのインポートに対応している Valhalla[5] というソフトウェアを使い、経路探索を行います。

まずインポート及び実行は簡単に処理をするためのスクリプトを用意しています。

Docker さえ動けば動作するので、お手元でぜひ試していただきたいです。

では、まずインポートをします。

```
git clone https://github.com/smellman/valhalla-docker-scripts.git
cd Valhalla-docker-scripts
cp ../data/planet-20220711.osm.pbf .
time docker run --rm -u `id -u`:`id -g` -e PBF_FILE=planet-220711.osm.pbf -v
${PWD}:/srv valhalla/valhalla:run-latest /srv/run_valhalla_all.sh
real    785m18.141s
```

13 時間ほどでインポートができました。

なお、Valhalla では最後にタイル状に生成したデータを tar ファイルで結合したものを経路探索のリソースとして使うのですが、そのファイルの生成時に瞬間的に 7GB/s ほどの書き込みスピードが出ました。

では、検索を行います。

```
docker run --rm -it -u `id -u`:`id -g` -p 8002:8002 -v ${PWD}:/srv valhalla/
valhalla:run-latest /srv/run_valhalla_api.sh
```

API サーバーが立ち上がったら、Valhalla のデモサイトを手元で動かします。

図 3 は手元で動かしている Valhalla でポルトガルからハンガリーまで経路探索をした例です。この経路探索が 600ms ぐらいのスピードで実現できます。

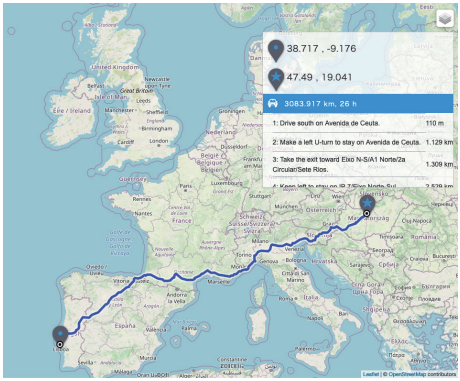


図 3: valhalla によるルーティング

5 https://valhalla.readthedocs.io

Planetiler による世界中のタイル作成

最後に実施したのは Planetiler[6] による世界中のタイル作成です。

このプロジェクトは OpenMapTiles プロジェクトの成果をベースとしつつ、一から Java で記述することで大幅な高速化を図ったプロジェクトです。

OpenMapTiles と比べると 50 倍以上速いというのがポイントです。ただ、50 倍以上速いのはメモリが 100GB 以上の環境の事を指しているようで、それよりも小さい、今回のような 16GB のマシンでは 200 倍ぐらい速い気がしています。

では、Planetiler 0.5.0 を使って計測をします。こちらはログに実測値が残ります。Runworld.sh では 20GB のメモリを指定しているため、swap メモリをあらかじめ確保しておく必要があります。

```
git clone https://github.com/smellman/planetiler-scripts.git
cd planetiler-scripts
wget https://github.com/onthegomap/planetiler/releases/download/v0.5.0/
planetiler.jar
./runworld.sh
Finished in 7h4m10s cpu:35h44m38s gc:1h9m2s avg:5.1
```

7 時間でインポートが終わりました。

なお、これは世界単位で 7 時間です。これはすごいスピードで、OpenMapTiles では日本の単位だけインポートするのに 2 日以上かかっていました。

ちなみに現在 OpenStreetMap Foundation Japan が運営しているタイルサーバー[7]ではこの Planetiler によって生成されたタイルを配信するようにしています。

今回検証したものが皆さんの OSM のデータを扱う第一歩になれば幸いです。

6 https://github.com/onthegomap/planetiler

7 https://tile.openstreetmap.jp/

松澤 太郎 (@smellman)

一般社団法人オープンストリートマップファウンデーションジャパンで技術担当をしています。他にも日本 UNIX ユーザー会会長、一般社団法人 OSGeo 日本支部理事など、地図と UNIX を愛するエンジニアとして活動中。

記憶するもののその先

伊藤 晋朗

我が家には 11 才の男の子がいる。名前は K 君だ。最近の彼はほぼ毎日 YouTube を見ている。何を見ているか確認すると 2 ちゃんねるのまとめを見ているようだ。そのせいで変な知識だけはどんどんつけてくる。そして、その影響かどうかわからないが今どきのゲーム機よりも、古いゲーム機が好きになってしまった。その結果、週末になるとやたらと「H ●● D OFF」に行きたがる。仕方がないので連れていくと、ジャンク品の棚のところでじーっと何かを見ている。基本は見ているだけで満足するが、そうかと思えばジャンク品として並んでいる昔のゲームのカセットを買ってきたりする。変な子に育ったもんだ。

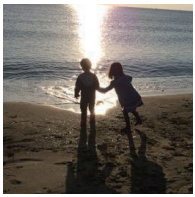
昔のゲームといえば、ある時からゲームにセーブ機能が付いていて、データを保存できるようになった。その頃はなんとなく、「ド●●エが楽」ぐらいに思っていたのを覚えているが、ゲームとして RPG やシミュレーションゲーム等には必須の機能だったのだろう。しかし、K 君が買ってきたジャンク品の古いゲームだと、結構、セーブデータが飛ぶことがある。K 君がとても悲しい顔をするので、部品が劣化してるのかなと思って、カセットを分解してみたら、直径 2cm ぐらいのボタン電池が入っていた。あの頃にデータを保存するっていうのは、ボタン電池のおかげだったという事をずいぶん経ってから知ることになった。昔の人はデータを保存するという事に、こんな方法を用意して、今とは違う戦いをしていたんだなと思って感慨深い。

そんな昔からデータを保存する方法というのが少しずつ進化して、半導体そのものにデータが記憶できるようになり、2000 年前後には、フラッシュメモリが入ったメモリカードが発売された。当時としては大容量なデータを保存できるという事で携帯電話やデジカメで使われていき急激に市場が拡大していった。その後、メモリの保存容量が増えるに従い、PC の HDD の置き換えとして SSD が発売された。耐衝撃性や OS 起動の速さが好評で、あっという間に市場が立ち上がり拡大していった。今ではデータセンター向けにも SSD が当たり前のように使われている。SSD の最初のインターフェイスは HDD と互換性のある SATA が使用されていたが、その後、SSD に最適なインターフェイスの規格として、NVMe™ という規格が策定された。この規格に対応した SSD を「NVMe SSD」と呼んだりするが、個人的に、アルファベットをそのまま並べただけの単語がどんどん長くなっていくのを見ていると、技術用語の新語というのは他に方法がないのだろうかと感じてしまう。

その NVMe 規格で議論されているコンセプト機能に「Computational Storage Drive（CSD）」と呼ばれるものがある。これはストレージ機能だけであった SSD に演算機能を持たせる製品のことである。データが保存されているメモリの近くで演算処理したら速くなるんじゃない?という事で検討されているものである。確かに CPU でデータを処理する

ためには、SSD から PCIe® を通って、DRAM にデータを展開して、CPU が演算して答えを出すので、SSD の中でデータを処理できるのであれば、無駄なデータ転送やデータコピーが発生しなくてよいのではというものである。このコンセプトはストレージの機能に関して、記憶するだけでなく演算能力を持たせようとしている。規格としてどうなるかは不明だが、現状のデータセンターの電力問題の対策への一つになるのかもしれないし、もしくは、日本の教育方法が「詰め込み教育」から移行しようとして、「ゆとり教育」として記憶するだけではなく思考する能力を育てようとしたのと同じように、何か現状ではまずいと思って方向転換を模索しているのかもしれない。

現在、義務教育中の我が家の K 君だが、買ったジャンク品をメ●●リで売って儲けようとしている。しかし、成功はしておらず、全く売れていない。YouTube のネタを覚えるだけでなく、考えている事は考えている。しかし、まだまだ成功するには遠い道のりだと思われる。今後も、私は温かい目で見守る予定だ。

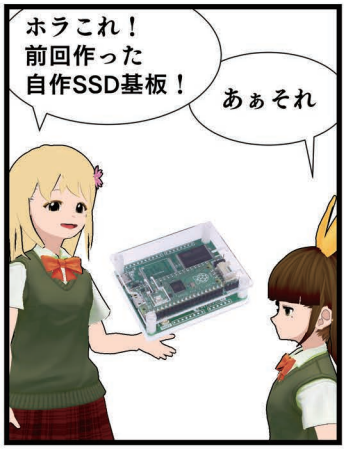


伊藤 晋朗（@ikainuk）

最近、町内会長として選挙の立会人の仕事をしてみた。ひたすら 12 時間以上座り続けるには座る姿勢が大事だ。世の中いろんな仕事があるもんだ。。。


かんがえるSSD
余熱
おーい、
磯野！
とある部室にて

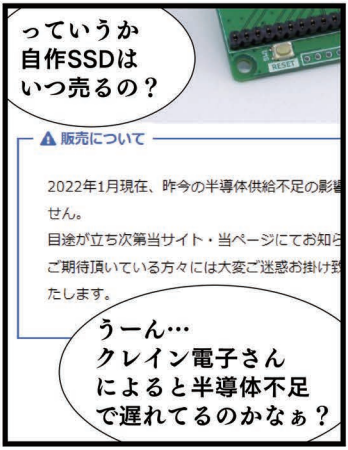

メモリコントローラ
自作しようぜ！
なんかどっかで
きいたセリフね
誰が磯野よ


ホラこれ！
前回作った
自作SSD基板！
あぁそれ

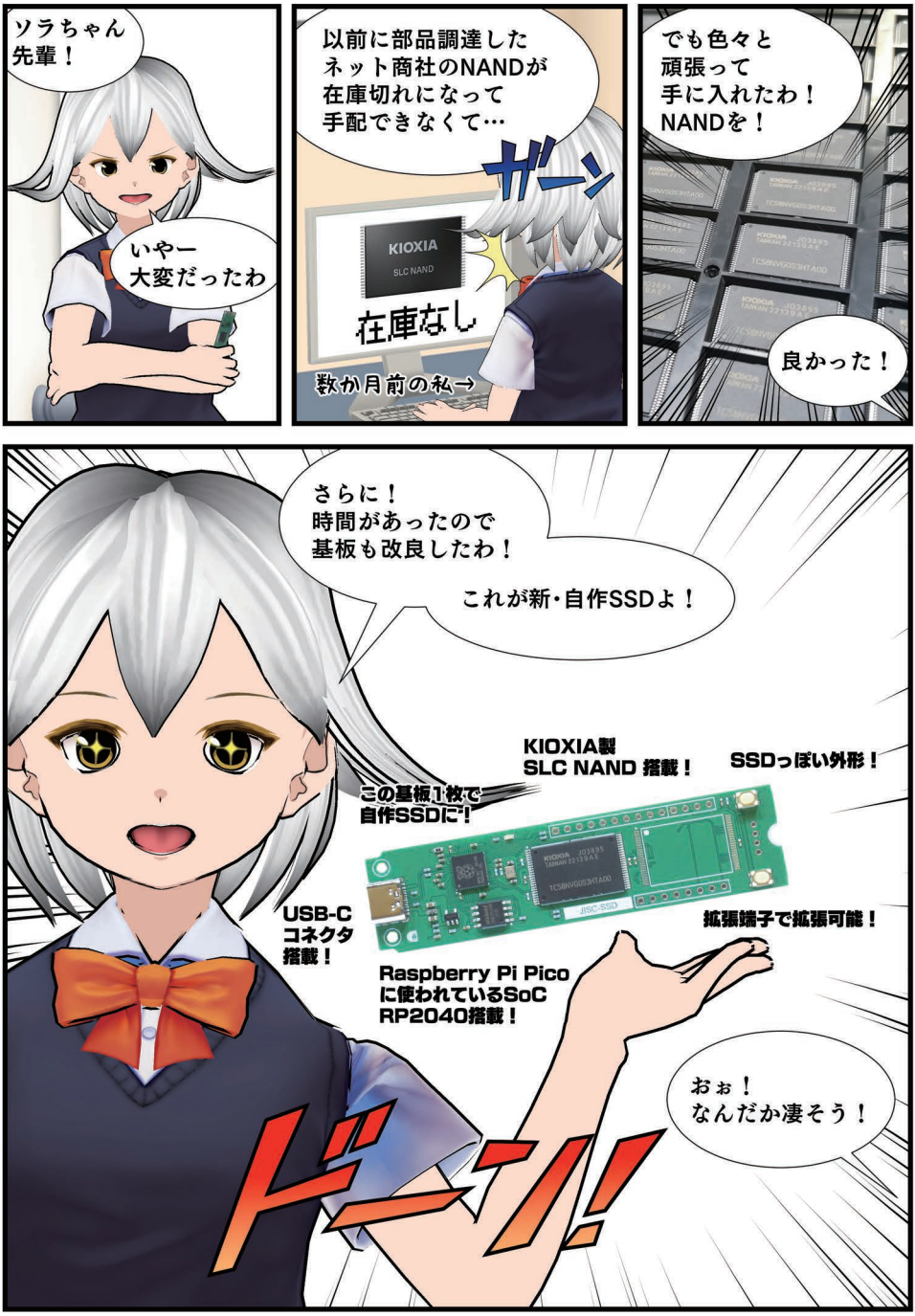

秋葉原で
展示されてた
みたいね
そうそう！
SSD同人誌
1号も
配って
もらったよ！


Maker Faire
Tokyo 2021は
オンライン開催
だったけど、
PDFの同人誌も
沢山ダウンロード
してもらえたみたい
SSD Doujinshi
ドヤァ…


なんか
ヨネミナー？
っていうのも
好評だった
みたいね


っていうか
自作SSDは
いつ売るの？
販売について
うーん…
クレイン電子さん
によると半導体不足
で遅れてるのかなぁ？


まぁ
なかなか難しい
ご時世だものね…
ちょっと
待った！


ソラちゃん
先輩！
いやー
大変だったわ
以前に部品調達した
ネット商社のNANDが
在庫切れになって
手配できなくて…
ガーン
KIOXIA
SLC NAND
在庫なし
数か月前の私→
でも色々と
頑張って
手に入れたわ！
NANDを！
良かった！
さらに！
時間があったので
基板も改良したわ！
これが新・自作SSDよ！
KIOXIA製
SLC NAND 搭載！
SSDっぽい外形！
この基板1枚で
自作SSDに！
USB-C
コネクタ
搭載！
拡張端子で拡張可能！
Raspberry Pi Pico
に使われているSoC
RP2040搭載！
おぉ！
なんだか凄そう！
ドーン！

* Computational Storage Drive

余熱 (@yone2_net)

Maker Faire Tokyo 2021 はオンライン開催でしたが、久々にヨネミナーをやることができ、局所的に話題にして頂けたので良かったです。なお、今回の漫画の作成にあたり妻に協力してもらいました。

In-Storage Computation 基板「JISC-SSD」

（株）クレイン電子 福屋 新吾

昨年、自作 SSD 基板「PicoSSD」を制作しました。自作 SSD の次なるバージョンとして、より SSD らしく、そして In-Storage Computing ができる拡張性のある基板を目指しました。

その名も Jisaku In-Storage Computation SSD、略して「JISC-SSD」です（図 1）。

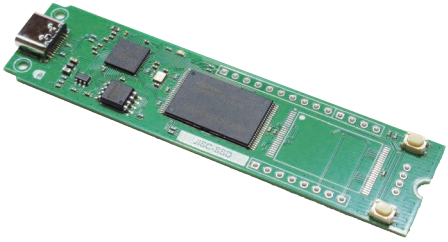


図 1: JISC-SSD 外観

「PicoSSD」との違い

PicoSSD は、Raspberry Pi Pico、または Airio-Base をホストとした、いわば "NAND フラッシュメモリ拡張基板 " でした。今回の JISC-SSD は Raspberry Pi Pico + PicoSSD の組合せを踏襲し、RP2040 ＋ NAND フラッシュメモリを一つの基板にまとめました。また、より SSD らしく見えるように M.2 の形状[1]に仕上げました。

USB コネクタは Type-C を採用しました。最近は 100 円均一ショップなどでも入手できるほどに普及してきましたので便利ではないでしょうか。その他のインターフェイスは、Raspberry Pi Pico 同様の LED(GPIO25) と、BOOT SEL スイッチ、これらに加え、RESET スイッチを搭載しています。ユーザー拡張用として 1 列ピンヘッダ端子を NAND フラッシュメモリチップの上下に配置しています。上部左側は GPIO 端子、その右側に I2C デバイス接続用端子、下側に UART と SWD 端子となっています。とりわけ下側の端子は Raspberry Pi の GPIO ピン配置と合わせており Raspberry Pi を用いたデバッグを意識した設計にしました（図 2）。

JISC-SSD は、Raspberry Pi Pico を踏襲した設計になっています。そのため、Raspberry Pi Pico の開発環境をそのまま用いることができます。

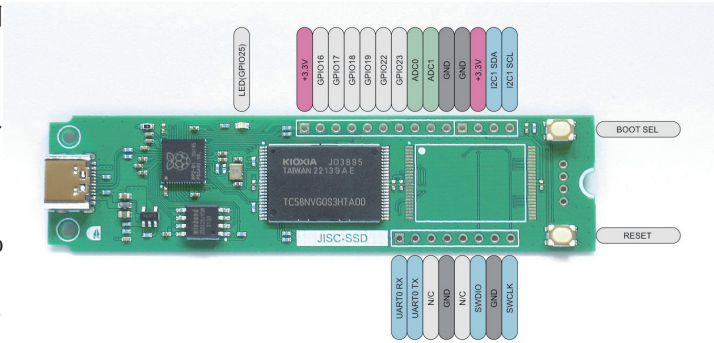


図 2: JISC-SSD ピンアサイン

1　M.2 コネクタへの接続はできません。

プログラミング

JISC-SSD は、Raspberry Pi Pico とある程度互換性を保った回路となっています。そのため、いくつかの Raspberry Pi Pico のサンプルコード[2]を修正なしに動かせます。そこで、実際に Raspberry Pi400 に Pico 公式のマニュアルに沿って開発環境をインストールし、定番の L チカプログラム「blink」をコンパイル、そしてそのまま何も考えずに JISC-SSD へファームウェアを転送しました（図 3）。すると見事に、JISC-SSD の LED が点滅をはじめました、成功です。つまり、JISC-SSD の開発をする上で新たに考える必要はなく、Raspberry Pi Pico ボードと同じ感覚で開発できるようになっています。

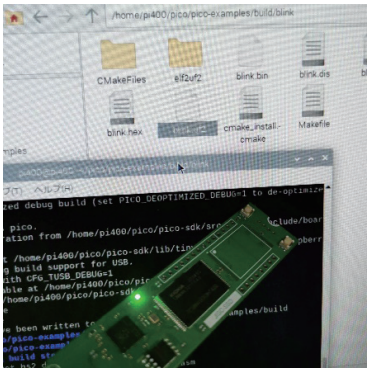


図 3: JISC-SSD の動作確認例

拡張性

JISC-SSD は、RP2040 のプログラムを実行可能で、ユーザー拡張端子を備えているため、様々な可能性を秘めている基板です。まさに、In-Storage Computation を手元で実現できるボードだと思います。例えば、I2C には温度センサーを、UART には無線 LAN や Bluetooth LE モジュールを接続し、マイコンは NAND フラッシュメモリに入っている Lua スクリプトを実行し制御するなど…そんなガジェットが 2280 サイズのワンボードに収まる、素敵な未来が待っているかもしれません。

入手方法

JISC-SSD の入手方法などの情報はクレイン電子 Web ページ[3]などを参照ください。準備が整い次第、各電子部品販売店での販売を計画しています。面白い使い方があったら SNS で投稿して頂けると嬉しいです。みんなで盛り上げていきましょう！！

2 https://github.com/raspberrypi/pico-examples
3 https://crane-elec.co.jp/

株式会社クレイン電子（@crane_elec） 福屋 新吾

基板の半田付けをする会社をしています。JISC-SSD の見た目良くないですか？自画自賛！

Raspberry Pi Pico SSD

村口

前回は、Mbed™ で動く SSD の記事を書かせていただきましたが、今回は、Raspberry Pi Pico で動く（予定の）SSD の記事を書きます。Raspberry Pi Pico には、大きな SRAM があるため、NAND フラッシュメモリの書き換え時のデータ退避先として有効活用できます。また、今回は、拡張ハミングコードによるエラー訂正検討についても説明します。

Raspberry Pi Pico SSD 構成

本章では、Raspberry Pi Pico SSD の構成について説明します。

ハードウェアおよび開発環境

ハードウェアは、株式会社クレイン電子さんの PicoSSD Raspberry Pi Pico 対応 SSD 学習ボード + Raspberry Pi Pico を使用させていただきました。

開発環境は、Raspberry Pi Pico ファームウェアをビルドするために、Raspberry Pi 4(Raspbian GNU/Linux™ 10) を使用し、CMake + 'g++' + OpenOCD で開発しました。Raspberry Pi と Raspberry Pi Pico 間は、SWD / シリアル接続をしています。

ソフトウェア

ソフトウェアは図 1 のような階層になっています。USB Mass Storage の基本機能については、TinyUSB[1] を利用させていただくことで実現しています。主に、RawNAND[2] 管理階層以下を作成しています。

- **RawNAND 管理**
  - アドレス変換処理
  - 初期不良ブロック探索
  - 代替ブロック管理
  - ライトキャッシュ処理
  - エラー訂正処理
- **RawNAND コマンド**
  - NAND IO 操作の組み合わせで RawNAND コマンドを発行する。
- **RawNAND 抽象レイヤ**
  - NAND IO 操作を GPIO 操作に関連付ける。

図 1: ソフトウェア階層

1 https://github.com/hathach/tinyusb/tree/4bfab30c02279a0530e1a56f4a7c539f2d35a293

2 素の NAND 型フラッシュメモリのチップ

Raw NAND メモリの構成と利用方法

本章では、Raw NAND メモリの構成、および利用方法を説明します。

Raw NAND メモリの構成

Pico SSD の NAND フラッシュメモリには、キオクシア製 TC58NVG0S3HTA00(SLC NAND 1Gbit 品) が使用されています。この NAND フラッシュメモリの 1 page は 2048+128[bytes] で構成され、64 page で 1 block という単位になります。メモリ全体は、1024 block で構成されています。リード操作、プログラム（ライト）操作は、page 単位未満での操作も可能ですが、消去操作は、block 単位の操作が最小単位となります。

Raw NAND メモリの利用方法

NAND フラッシュメモリの不良は、block 単位で発生します。データシートによると、ライフタイムで 20 block の不良が発生するとのことなので図 2 のような block 割り当てとしました。Block 管理テーブルには、1020 番目～ 1023 番目の 4 block 分を割り当てており、1020 番目の Block を最初に参照します。もし 1020 番目が故障すれば、次は 1021、1022、1023 番目と参照先を変更していく考え方で使用します。なお低確率ですが、運悪く、最初から Block 1020、1021、1022、1023 番が全て壊れている場合は、いきなり使用不可となります。NAND フラッシュメモリの工場出荷時に確実に使用できることが保証されている Block 0 を代わりにご活用いただくなど工夫は可能ですのでご参考まで。

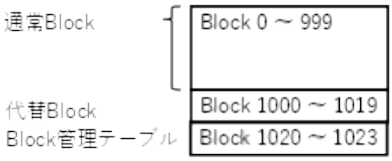


図 2: block 割り当て

Block 管理テーブルは、各 block の先頭 page に図 3 のようなフォーマットで記載することにしました。Marker code で、破損状態を読み取ることとします。各 Block address は、0 ～ 1023 の値は有効なアドレスとし、0xffff 等、それ以外の値は、無効なアドレスとします。各 Block address の整合性も含めて確認したい場合は、全体のハッシュ値、CRC 等を保持するフィールドを定義しておくとよいでしょう。

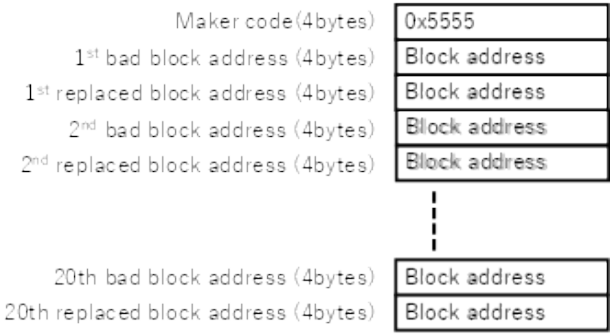


図 3: 各 block の先頭 page に記載するフォーマット

エラー訂正

本章では、拡張ハミングコードを用いたエラー訂正について説明します。

拡張ハミングコードの適用検討

拡張ハミングコードは、情報に検査ビットを加えて、1bit の誤り訂正と 2bit の誤り検出が可能なエラー訂正手法で、古くから ECC メモリ等で利用されてきました。拡張ハミングコードに必要な検査ビット数 [bits] は、転送情報ビット数 [bits] から、次の式で表すことができます。

```
検査ビット数 [bits] = roundup ( log2 ( 転送情報ビット数 [bits] + 1 ) ) + 1
```

この式の転送情報ビット数は、どのビットが間違っているかを示すための情報量になります。その後の +1 は、転送が正しくできたケースを区別するための情報量です。最後の +1 は転送情報の全 bit を XOR（排他的論理和）した検査結果を格納する情報量です。この検査ビットによって 2bit の誤りが検知可能になります。横軸に転送情報ビット数、縦軸に拡張ハミングコードに必要な検査ビット数をとると、図 4 の青線になります。転送したい情報がちょうど 2 のべき乗になったタイミングで、必要な検査ビット数が増加します。転送情報を増やすにしたがって、必要な検査ビットの増え方は、次第に、緩やかになります。

一方、今回使用する NAND フラッシュメモリは、1page あたり 2048[bytes] につき、128[bytes] の検査ビット用スペースが用意されており、つまり検査ビット量は、情報量の 1/16 以下の大きさに制限されることになります。よって、図 4 の赤線よりも下側の領域で検査ビットを用意する必要があります。この条件で拡張ハミングコードの適用を検討すると、転送情報 256bit につき、10bit の検査ビットを付けるという使い方が適用可能と判断できます。

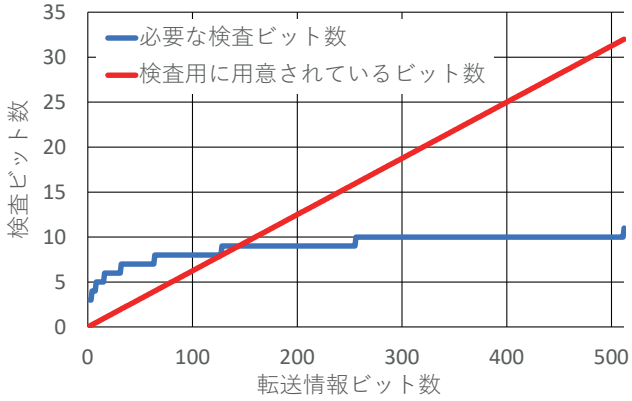


図 4: 転送情報ビット数と拡張ハミングコード

バーストエラー対策案

拡張ハミングコードは、256bit+10bit の塊の中で、1bit のエラー訂正と 2bit のエラー検出が可能ですが、256bit+10bit をそのまま固めて配置すると、2bit 以上のバーストエラー[3]を修正することができません。もちろん、バーストエラーが出やすいかどうかは、そのメモリの性質によるのですが、仮に連続したバーストエラーが出やすいと仮定した場合に、どういう対処ができるか事前に考えておくことにします。

例えば、1page には、256bit+10bit の拡張ハミング符号語を 64 個格納することができます。まず、この 64 個の bit0 だけを取り出して、64 個連続して並べます。次に、符号語から bit1 だけを取り出して、64 個並べます。という並べ方をすると、64 個と、次の 64 個の境界でバーストエラーが起きた場合は、やはり対処できませんが、一部のバーストエラーはエラー訂正で対応可能になります。

おわりに

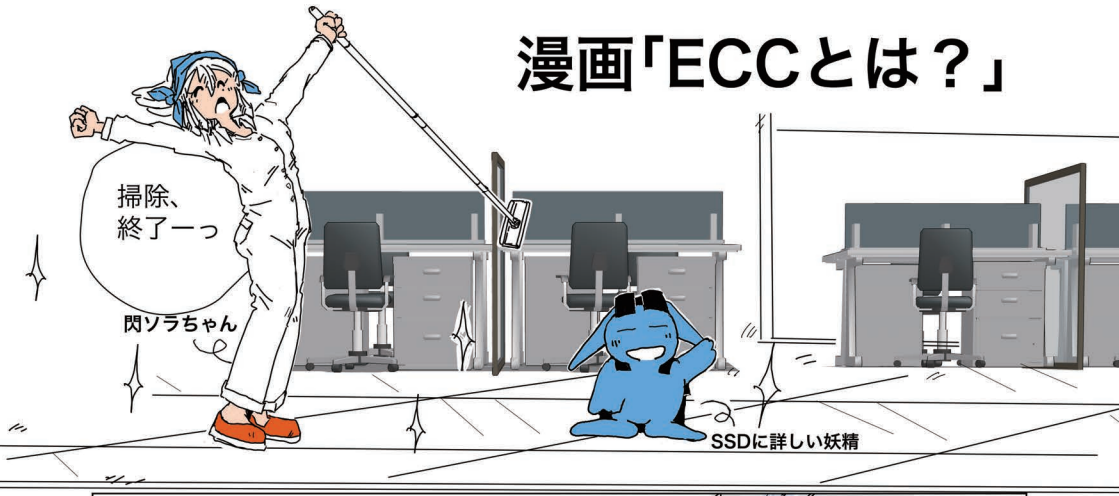
Raspberry Pi Pico SSD およびその ECC の考え方について説明させていただきました。キオクシアでは、一部でこういった草の根趣味活動のようなものが行われている面白い会社なのですが、私はすでに 2022 年 4 月末でキオクシアを退職しており、5 月からエネルギー業界のスタートアップで働いています。キオクシア・東芝時代から、社内外含め本当に多くの方にお世話になりました。この場をお借りして、感謝申し上げます。当時の上司や同僚から役に立つご助言を数多く頂きましたが、特に印象に残っている言葉があります。村口が新入社員だった当時の上司、中田さんから頂いた言葉です。私が当時、秋葉原に通っていることを伝えたところ、「見たら、買え！！」とおもしろ可笑しくアドバイスしてくれたのでした。機を逃して後悔するぐらいなら、思い切った行動を取るべしという意です。この無鉄砲とも言える助言に沿うように？私の残りの人生を、エネルギー問題を解決することに使っていこうと決めました。いままで半導体開発しかしてこなかった自分にとっては大きな転換です。そして、この決断を実際に実行に移すことができたのは、理解のある妻のおかげです。いつもありがとう。

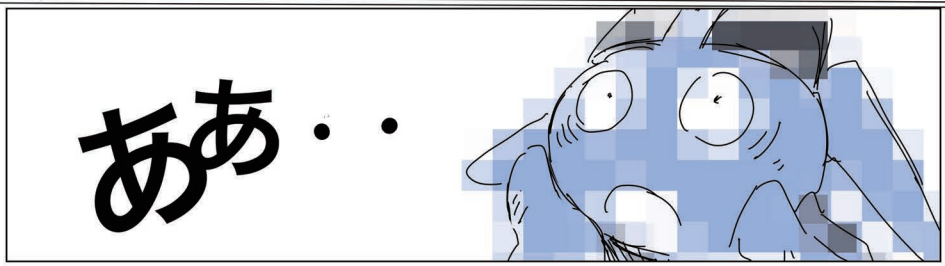
3　データを転送する際に連続して発生するビット誤りのこと

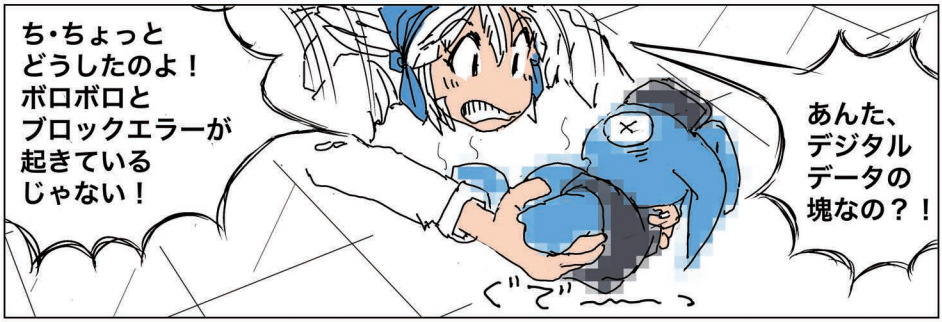
村口

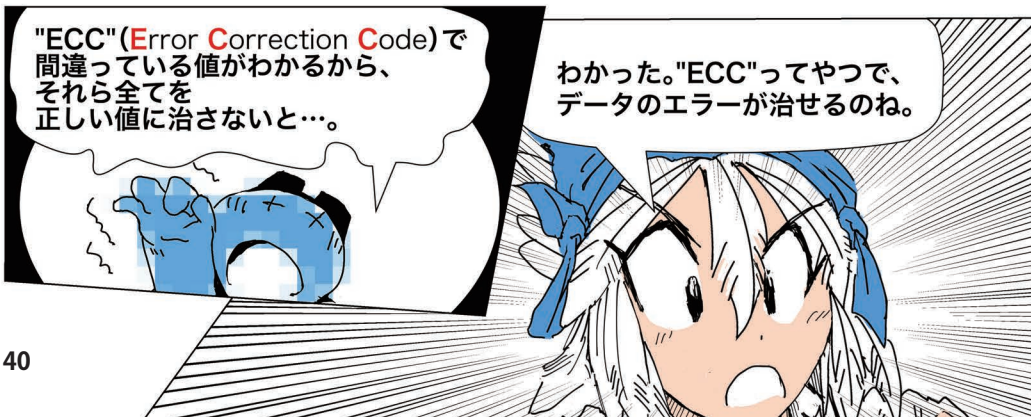
休みの日は娘と散歩（大船フラワーセンターにて）

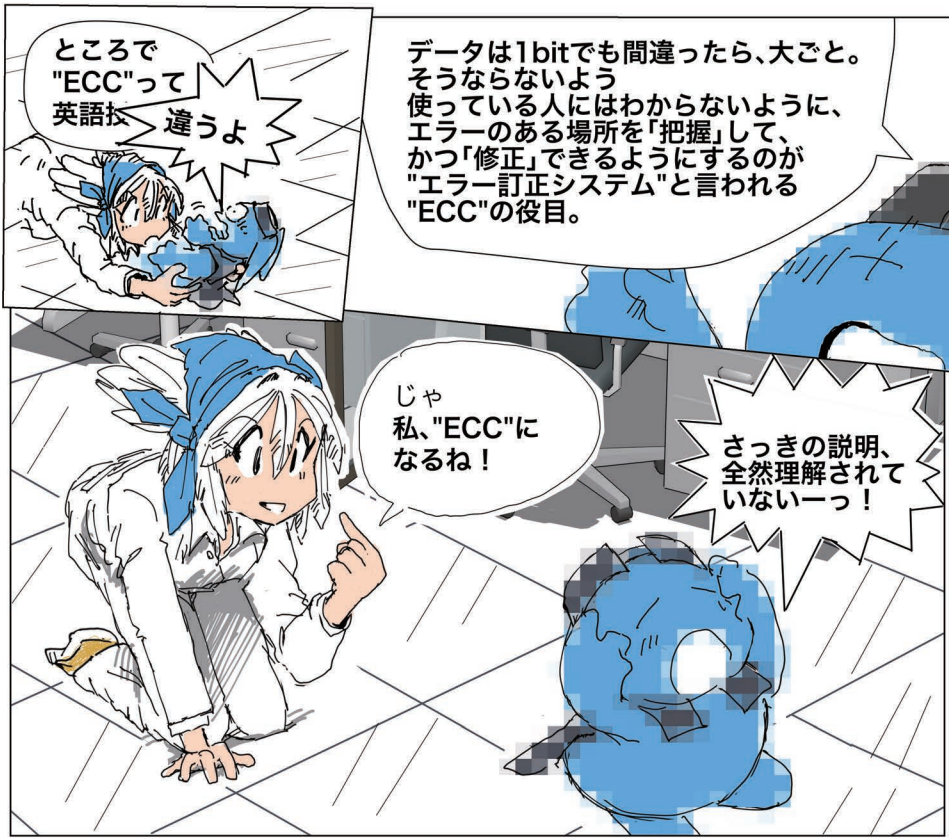
漫画「ECCとは？」

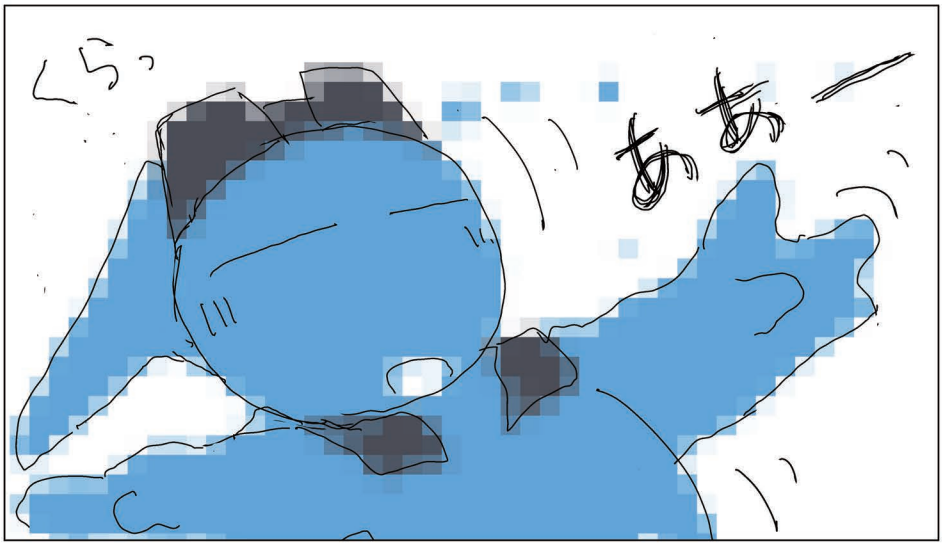

掃除、
終了ーっ
閃ソラちゃん
SSDに詳しい妖精


ああ・・

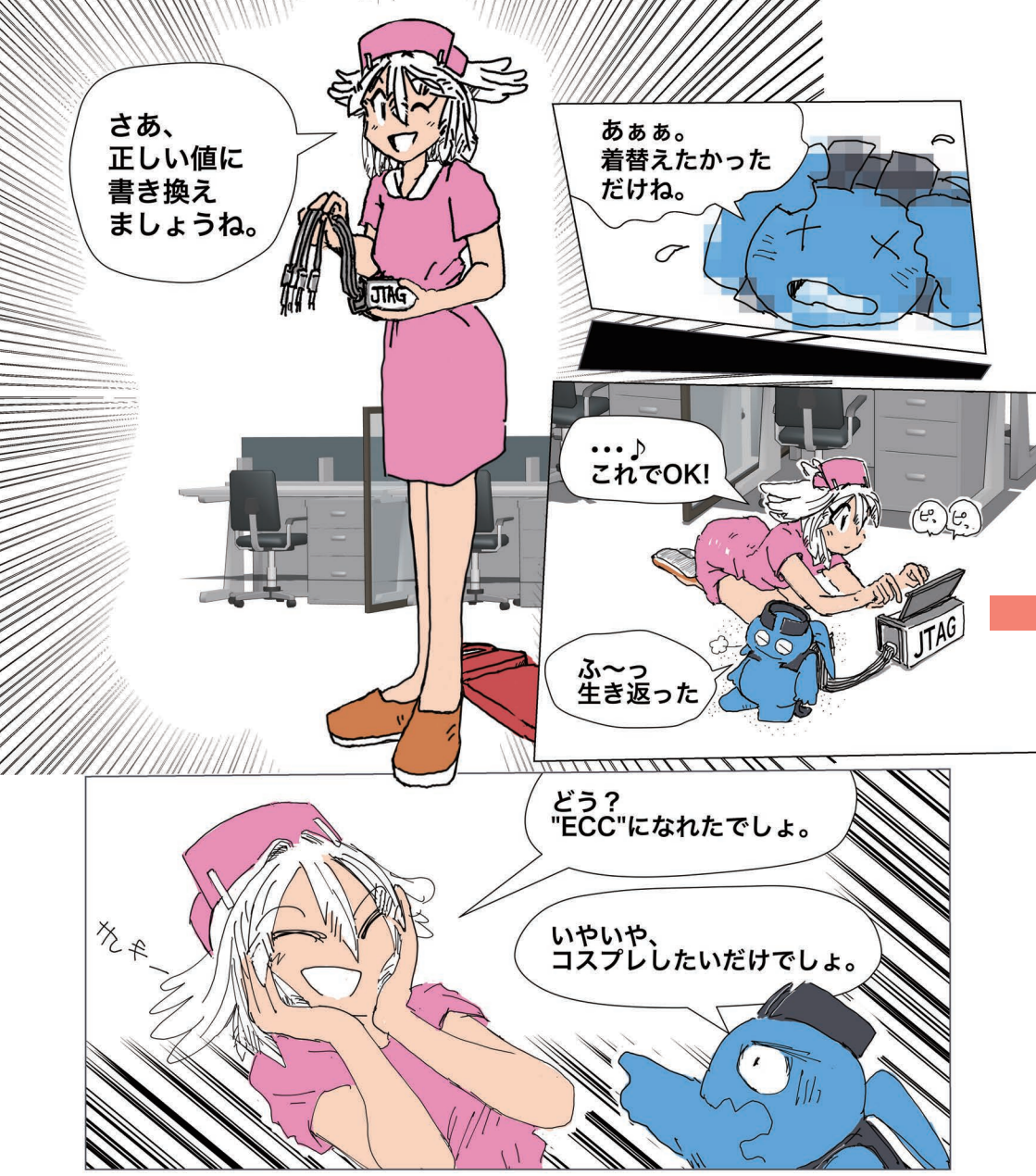

ち・ちょっと
どうしたのよ！
ボロボロと
ブロックエラーが
起きている
じゃない！
あんた、
デジタル
データの
塊なの？！
ぐでで〜っ


"ECC"(Error Correction Code)で
間違っている値がわかるから、
それら全てを
正しい値に治さないと…。
わかった。"ECC"ってやつで、
データのエラーが治せるのね。


ところで
"ECC"って
英語だ
違うよ
データは1bitでも間違ったら、大ごと。
そうならないよう
使っている人にはわからないように、
エラーのある場所を「把握」して、
かつ「修正」できるようにするのが
"エラー訂正システム"と言われる
"ECC"の役目。
じゃ
私、"ECC"に
なるね！
さっきの説明、
全然理解されて
いないーっ！


くらっ
あぁ


ガッ
ECC
ちょっ…。
ちょっと待ってよ！
今、用意するから。
んっ！
"ECC"の引き出しって何？！
しかも
"AED"を魔改造して
"Error Cancellation Costume"
(エラー・キャンセレーション・
コスチューム)って！
AED
Error
Cancellation
Costume
じぇーん
大丈夫なん？
エラー訂正システム
パカ
装着



じむ (@Hirameki_Sora)

別会社に勤めておりますが、「薄い本を作る」ということで、今年もお呼ばれされました。

閃ソラちゃんも、ここでしかお目にかかれないようになりましたが、忘れ去られないように続けて描いていこうと思います。

ECC でデータの誤りを訂正してみた

川口 優樹

最近、軽量なノート PC（1kg 程度の重量）を持って、カフェなどで作業する方をよく見かけます。この軽量なノート PC を実現しているキーアイテムに SSD があります。SSD はこれまでデータの保存に使用されていた HDD よりも軽量で小型という利点があります。

良いことづくしな SSD ですが、欠点もあります。それは放置することでデータが化けてしまうことです。化けてしまったデータは化けている箇所を特定し、訂正する必要があります。

SSD ではデータに「誤り訂正符号 (ECC : Error Correction Code)」と呼ばれるヒントを付け、このヒントをもとに化けてしまったデータを訂正しています。今回は ECC をもとにして化けたデータを訂正するしくみを簡単に紹介します。

SSD の構造と動作

一般的な SSD は図 1 に示すように大きく分けて 2 つの部品から構成されています。

1 つ目はコントローラです。コントローラはコンピュータからデータや命令を受け取り、フラッシュメモリに対してデータを読み出し・書き込みを行います。2 つ目はフラッシュメモリです。フラッシュメモリはコントローラから受け取ったデータを格納する場所です。

コントローラを介して、フラッシュメモリへデータを書き込み、あるいはフラッシュメモリからデータを読み出すことで、SSD はデータを記憶しています。

SSD は 1 日の間に数 GB から数 TB 程度の読み書きをしていますが、たまにフラッシュメモリから読み出したデータが化けていることがあります。

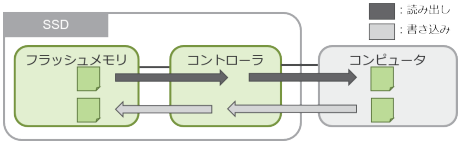


図 1: 一般的な SSD の構造

読み出したデータが化ける

デジタルデータは「0」と「1」がたくさん集まって構成されており、フラッシュメモリでは、図 2 に示すようにメモリセルに電子を蓄えることでデータを保持しています。ところが電子は軽い粒子なので、ほっておくと自由に飛び回ってしまいます。そのため、時間経過とともにメモリセルから電子が抜けてしまいます。

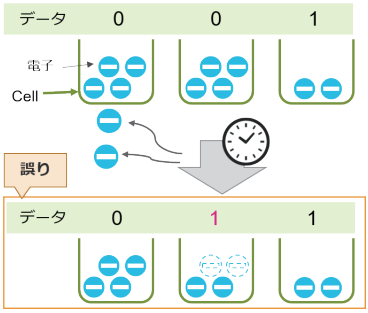


図 2: メモリセルと誤りの様子

電子が抜けてしまい、0/1 を正しく記憶することができなくなると、書いた時とは異なるデータに化けてしまいます。このデータが化けた状態を「誤り」と呼びます。

誤りをそのままコンピュータに送るわけにはいかないので、コントローラがデータを読み出す際に、誤りを検出し、訂正しています。

誤りを検出してみる

ではどうやって誤りを検出するのでしょうか。以降では、話を簡単にするため、図 3 に示す 2bit のデータのうちいずれかを入力するものとし、誤りは「0 から 1」あるいは「1 から 0」の 2 パターンとします。さらにデータは図 4 のような雑音の多い通信路を通るものとし、この通信路を通ったデータは 1bit の誤りが生じるものとします。なお、通信路の両端はそれぞれフラッシュメモリへの書き込みと読み出しに対応しています。

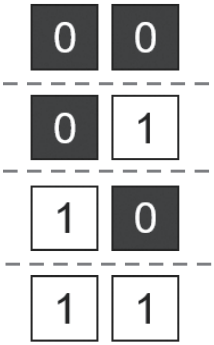


図 3: 2bit のデータ

誤りを検出するために、データが誤っていることを表すヒントをデータへ付与してみます。今回は図 5 に示すルールにもとづき、2bit のデータに対して 1bit のヒントを付与することにします。例えば入力データを "01" というデータとします。"01" はルール (b) が該当するため、ルール (b) からヒントとして "1" を付与し、"011" を作成します。このヒントを付与したものを「符号語」と呼びます。では、この符号語を実際に通信路に通して、誤りを検出できるか確認してみましょう。

図 6 に示すように "011" を通信路に入力したところ、"001" が出力されたとします。ここで出力 "001" の先頭 "00" に付与されるヒントは、本来は図 5 のルール (a) にあるとおりであって、"1" ではありません。このことから誤りの発生を検出できます。

このように入力データに対して 1bit のヒントを追加することで、1bit の誤りを検出することができました。以上のような誤りを検出するために付与する符号を「誤り検出符号 (EDC: Error Detecting Code)」といいます。

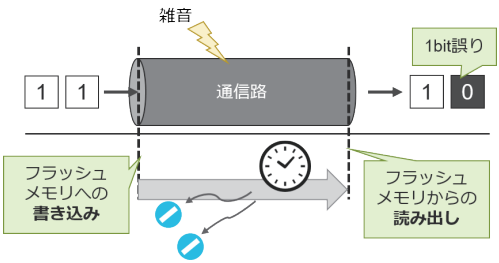


図 4: 雑音の多い通信路

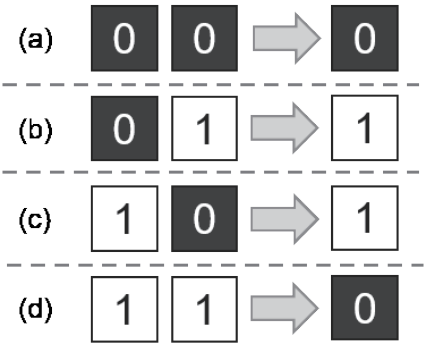


図 5: 使用するルール

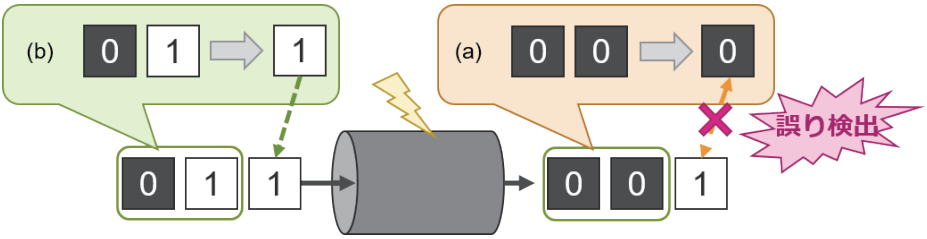


図 6: 誤りを検出する様子

しかし今回の例では出力された 0/0/1 のどれが誤りかがわかりません。そこで 2bit 以上のヒントを追加することで、誤りを訂正できないか考えてみましょう。

誤りを訂正してみる

前節の「2bit 以上のヒントを入力データに追加することで誤り訂正できるか」を動機づけとして、今回は誤り検出用に作成した 1bit のデータと 2bit の入力データの合計 3bit をヒントとして入力データに付与することにしてみます。具体的には、入力データを "01" とすると、図 5 に示したルール (b) から "1" が得られます。これの後ろに入力データ "01" を付けた "101" を改めてヒントとします。そして、入力データ "01" にヒント "101" を付与した "01101" を符号語としてみます。

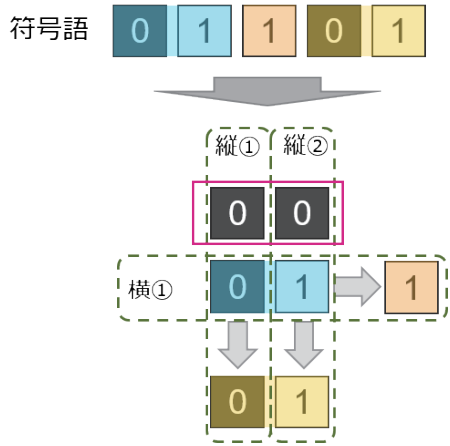


図 7: 誤り訂正時のルール

続いて図 7 のように符号語を配置して、縦と横でそれぞれ図 5 に示すルールに従っているかを確認します。なお、図中の "00" は読み出し時に通信路の外で付与するもので[1]、これに誤りが生じることは無いものとします。また "01" の "1" を 1bit 目 (図 7 の縦②に対応)、"0" を 2bit 目 (図 7 の縦①に対応) と呼ぶこととします。

では、1bit の誤りを含む符号語の場合、どのように訂正されるかを見てみましょう。

いま、通信路から符号語 "11101" が出力されたとします。この符号語を前述の図 7 の手続きに従って 図 8 左の (1) のように配置します。次に、(1) の横①がルールに従っているかを確認します。(1) の青塗りの部分は "11" なのでルール (d) に該当します。ルール (d) からは "0" が得られますが、横①では "1" となっているためルールに従っていません。以上から、この符号語に誤りがあることが確認できました。

1　図 7 で入力データ “01” の上に黒マスの “00” が付与されていますが、入力データが何であれ縦①と縦②において図 5 のルール (a) あるいは (b) に従った結果得られる数値が、入力データと同じになること (ここでは “01” と同様に縦①が “0”、縦②が “1” となっていること) がミソです。

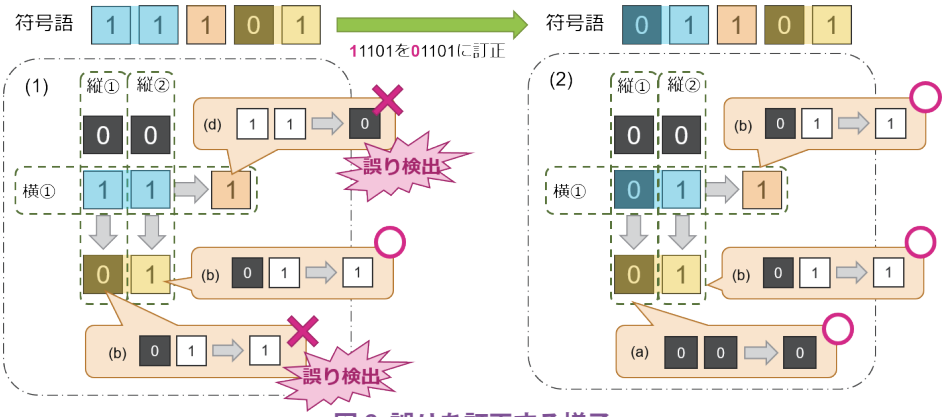


図 8: 誤りを訂正する様子

次に、(1) の縦①と縦②のそれぞれがルールに従っているかを確認します。まず縦②はルール (b) に従っているので、1bit 目には誤りが無いとわかります。しかし縦①はルール (b) に従っていないため、2bit 目に誤りがあることが確認できます。

以上から、符号語 "11101" の先頭の "1" に誤りがあるため、これを "0" に訂正します。訂正した符号語 01101 を再度同様の手続きで図 5 のルールに従っているかを確認します (図 8(2))。すると横①・縦①・縦②はそれぞれルール (b)・(a)・(b) に従っているため、訂正後の符号語 "01101" には誤りが無いことが確認できました。以上から入力データに 2bit 以上のヒントデータを追加することで、1bit の誤りを訂正できました。このように、誤りを検出し、かつ訂正することができるヒントデータが ECC に該当します。

おわりに

いかがでしたか。今回は誤り検出を縦横で行うことで誤りの位置を特定し訂正する様子をご紹介しました。誤り検出や訂正技術は SSD の他に通信分野でも使用されている技術で、現代生活になくてはならない存在になっています。この記事がきっかけとなり、誤り検出や訂正技術に興味を持っていただけると嬉しいです。

川口 優樹
昔からコンピュータをいじるのが好きで、
気づいたら SSD の開発に携わっていました！

え！音が鳴る SSD でコンピューティングを！？

にちか

２２年４月頃、本書の企画会議がありました。

Pochio さん「にちかくん、君の“人力メモリ書き込み大会”という提案はイマイチだけど、こっちの“音が鳴る SSD”は面白いよ！」
余熱さん「自作 SSD に FM 音源とか積んだら良さそう！」
Pochio ＆余熱「「ちなみに今年の記事は、In-Storage Computation でよろしく！」」
にちか「・・・・・・・」
にちか「出来らぁっ！」

出来たもの：MIDI Kioku Xfer（トランスファー）

MIDI 楽器の演奏を録音する際、PC へ接続し、専用ソフトを起動しなければなりません。また、MIDI 演奏データの再生も、多くの場合 PC が用いられます。ここでもし、PC の代わりに録音・再生を単独で行えるストレージデバイスがあったならば、それは In-Storage Computation の一形態と言えるのではないでしょうか。

そこで作ってみたのが、MIDI Kioku Xfer（トランスファー）（略して、MKX）です。MKX の外観を図1に示します。MKX は、SSD であると同時に、MIDI 音源でもあるため、演奏データの録音と再生が可能なデバイスです。つまり、MKX を MIDI 楽器に接続すれば、演奏しつつ録音することができます。また、演奏データは NAND フラッシュメモリに保存されるため、いつでも再生することができます。さらに PC へ接続すれば、保存された演奏データを編集することも可能です。以上から MKX は、MIDI 楽器に対するデジタルカメラやボイスレコーダーのようなものとご想像ください。

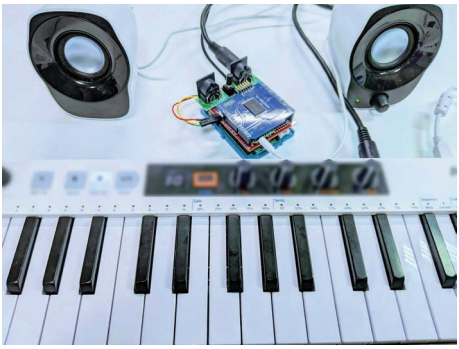


図 1: MKX の外観

MKX の構成

本章では、MKX のハードウェアとソフトウェアの構成について説明します。

ハードウェア構成

ハードウェアを構成するパーツを図 2 に示します。MKX は、Airio-Base（緑の基板）、MIDI I/F シールド（赤の基板）と、SLC NAND シールド（青の基板）から構成されます。I/F シールドは、MIDI 信号の送受信回路を持ちます。また、演奏データを音声出力するため、YMF825[1] を備えています。なお、YMF825 は SPI 信号で制御しますが、Airio-Base にはピンの割当がないため、SD カードスロットのピンを引き出しています。ちなみに、Airio-Base の代わりに JISC-SSD を用いる場合は、ユーザー拡張端子に SPI ピンを備えているため、このワークアラウンドは不要です。

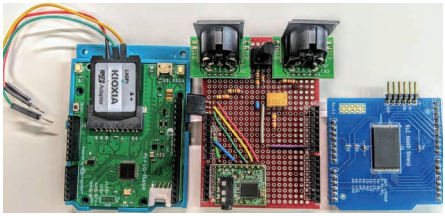


図 2: MKX の構成パーツ

ソフトウェア構成

MKX のソフトウェア構成を図 3 に示します。MKX は 4 つの OSS ライブラリ、1 つの自作クラスと、メインプログラムから構成されます。Mbed™ ライブラリ[2] は、タイマー機能、ピンの I/O 機能や割り込み機能を提供します。MIDI ライブラリ[3] は、MIDI 信号の受信と受信時のハンドリング機能を提供します。YMF825 ライブラリ[4] は、YAMAHA が公開した Arduino 用ライブラリを Mbed へ移植した物で、YMF825 モジュールのドライバ機能を提供します。USBMSD[5] は、クレイン電子が公開しているライブラリで、NAND フラッシュメモリを R/W し、ホストに対し SSD として振る舞う機能を提供します。SMF クラスは、Standard MIDI File フォーマットへのエンコード / デコード機能を提供する自作クラスです。メインプログラムは、以上の機能を組み合わせて、MIDI 信号の録音 / 再生機能を提供します。

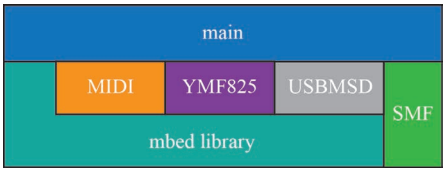


図 3: MKX のソフトウェア構成

1 YAMAHA 製 FM 音源 LSI(https://akizukidenshi.com/catalog/g/gM-12414/)
2 https://os.mbed.com/users/mbed_official/code/mbed/builds/65be27845400
3 https://os.mbed.com/users/okini3939/notebook/midi/
4 https://github.com/hasebems/YMF825_sample
5 https://github.com/crane-elec/USBMSD_step1

録音機能の実現

本章では、MKX の録音機能の実装について説明します。

ファイルシステム

以下、MKX を FAT32 でフォーマットした場合の仕組みを説明します。あるストレージのトップディレクトリ（ルートディレクトリ）上に存在するファイルは、ルートディレクトリエントリー（RDE）セクタのテーブルで、ファイル名やサイズ、保存先といった情報が管理されます。つまり PC は RDE セクタを参照することで、どこにどんなファイルが有るかを把握します。MKX が演奏を録音した場合、そのデータを PC に見つけてもらうには、RDE セクタを正しく編集しなければなりません。それは流石に大変なので、MKX に以下の制約を設けることで、RDE セクタの編集操作を回避しています。

1：MKX は常に単一の録音データを保存する

2：録音データのファイルサイズやファイル名も固定

ちなみに SMF フォーマットはサイズが可変なので、末尾の領域は 0 でパディングします。

SMF フォーマット

SMF フォーマットの構造を図 4 に示します。SMF は図 4 の通り、ヘッダチャンクとトラックチャンクの集まりです。さらに各トラックチャンクには、いつ、どの音を鳴らすかという情報が集まっています。MKX は、演奏を録音する際にこのファイル構造を組み立てる（エンコードする）必要があります。そこで、図 4 のヘッダチャンクから Length フィールドまでをヘッダパート、Event # 1から #n を演奏パート、End of track 以降をフッタパートと考えます。この 3 つの内、演奏パート（いつ、どの音を鳴らすか）のみをエンコードすることにしました。

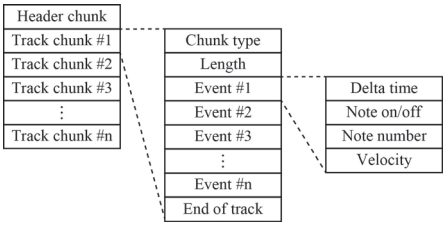


図 4: SMF フォーマット

例として、MKX でチャルメラのメロディーを録音した時のバイナリダンプを図 5 に示します。図 5 より、演奏パート（青色の範囲）に “ ドレミレドドレミレドレ ” が保存できていることがわかります。

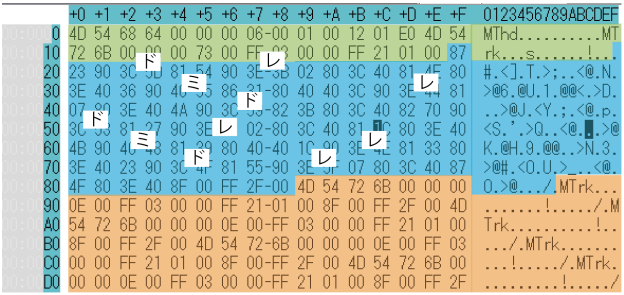


図 5: SMF のバイナリダンプ

再生機能の実現

音が鳴って録音ができるなら、再生もしたいですよね。ということで録音した演奏データの再生機能を実装しました。

再生機能のシーケンス図を図 6 に示します。USBMSD クラスで NAND フラッシュメモリから演奏データを読み出し、SMF クラスでデコードし、YMF825 ライブラリで音を鳴らします。この時、音を鳴らすタイミングは、Mbed ライブラリのタイマー割り込みで制御しています。割り込み発生時に音を鳴らし、さらに次のタイマー割り込みをセット、とバケツリレーのように割り込みを繰り返すことで、演奏データを再生できます。再生する音のタイミングが録音した時とわずかにずれてしまうのですが、これはデルタタイムの計算で整数除算を行っており、小数点以下の値の切り捨てが起きているためです。

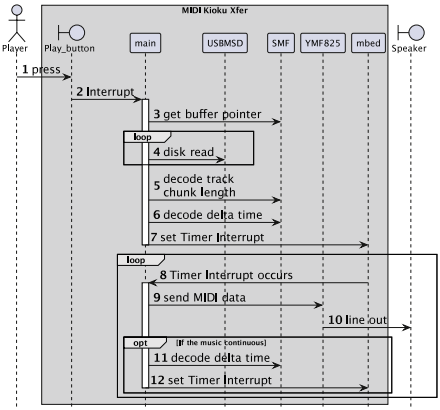


図 6: 再生機能のシーケンス図

MKX のデモ

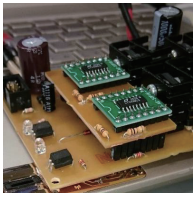
MKX のデモ動画を撮影しました。
リンク を掲載いたしますので、是非ご覧ください。

おわりに

当記事では MIDI Kioku Xfer を製作し、その実装について説明しました。また、MIDI 演奏データの録音と再生デモを行うことが出来ました。

製作当初は、何をもって In-Storage Computation と言えば良いのか、そもそも SSD から音が鳴って喜ぶ人は居るのか、と大変悩みました。一方で、次世代の SSD を自作しようという意気込みには非常に共感し、とても胸が踊りました。

当記事の執筆にあたって、偉大な先人達の作品・知見をいくつもお借りしました。この場を借りて御礼申し上げます。



にちか (@lxacas)
電子音楽歴 0 年。
音が鳴るコンピュート SSD に挑戦しました。
フォトカプラはてきとーに買っちゃだめ。
激安品が正常動作せずに 2 週間悩みました。

- 記憶容量：1MB（1 メガバイト）＝1,000,000(10 の 6 乗) バイト、1GB（1 ギガバイト）＝1,000,000,000（10 の 9 乗）バイト、1TB(1 テラバイト）＝1,000,000,000,000(10 の 12 乗) バイトによる算出値です。しかし、1GB=1,073,741,824 (2 の 30 乗) バイトによる算出値をドライブ容量として用いるコンピューターオペレーティングシステムでは、記載よりも少ない容量がドライブ容量として表示されます。ドライブ容量は、ファイルサイズ、フォーマット、セッティング、ソフトウェア、オペレーティングシステムおよびその他の要因で変わります。使用可能なストレージ容量（さまざまなメディアファイルの例を含む）は、ファイルサイズ、フォーマット、設定、ソフトウェア、Microsoft オペレーティングシステムやプリインストールされたソフトウェアアプリケーションなどのオペレーティングシステム、またはメディアコンテンツによって異なります。実際のフォーマット済み容量は異なる場合があります。
- 画像は説明用です。実際の商品・サービスとはデザイン・仕様が一部異なる場合があります。
- 「2.5 インチ」は SSD のフォームファクターを示し、SSD 自体の外形寸法を示すものではありません。
- NVMe は、NVM Express, Inc. の米国またはその他の国における登録商標 または商標です。
- PCIe は PCI-SIG の商標です。
- YouTube は、Google LLC の商標です。
- USB Type-C は、USB Implementers Forum の商標です。
- Raspberry Pi 、Raspberry Pi Pico は、Raspberry Pi Foundation の商標です。
- Mbed は、Arm Limited（またはその関連会社）の米国等での登録商標または商標です。
- Linux は、米国およびその他の国における Linus Torvalds の商標です。
- Intel および Core は、Intel Corporation またはその関連会社の商標です。
- PostgreSQL は、PostgreSQL Community Association of Canada の商標または登録商標です。
- Prometheus は、米国およびその他の国における The Linux Foundation の商標です。
- Docker は、アメリカ合衆国およびその他の国における Docker, Inc. の商標または登録商標です。
- Java は、オラクルおよびその関連会社の商標または登録商標です。
- その他記載されている社名・商品名・およびサービス名などは、それぞれの会社が商標として使用している場合があります。
- 製品の仕様、サービスの内容、お問い合わせ先などの情報は 2022 年 9 月時点の情報です。予告なしに変更される場合がありますので、あらかじめご了承ください。ここに含まれる技術およびアプリケーション情報は、最新の該当するキオクシア製品仕様が対象となります。

本同人誌に掲載の情報や内容については十分に注意を払っておりますが、その利用によって利用者にいかなる損害や被害が生じましても、一切の責任を負いません。必ず利用者ご自身の責任においてご利用ください。

本同人誌記載の部品性能は部品単体での性能であり、自作 SSD の動作、性能等を保証するものではありません。

宮内

前巻のとき「また面白いことやってますね！」と言っていたら、今回お誘いいただいたので参加。新キャラに便乗していく。いつかSSDでもカードゲームを作れるといいなぁ。

■ SSD Doujinshi 2 - SSDの同人誌 2

2022年 9月 3日 第1版第1刷発行
2022年 11月 1日 第1版第2刷発行
著者：ragnag / とだ勝之 / Pochio / さしすせそ / 松澤 太郎 / 伊藤 晋朗
余熱 / 福屋 新吾 / 村口 / じむ / 川口 優樹 / にちか / 宮内
表紙イラスト：とだ勝之
表紙デザイン：ホーリー
編集：余熱 / Pochio / にちか / みおりん / あおいさや
発行：キオクシア株式会社
連絡先：kioxiahq-Exhibition-SSD@kioxia.com
印刷：株式会社 プリントパック

KIOXIA

キオクシア株式会社

〒 108-0023　東京都港区芝浦 3-1-21 田町ステーションタワー S

www.kioxia.com