



Technology Consulting Company  
Research, Development &  
Global Standard

# BitVisorにおけるUSBの仮想化 方式

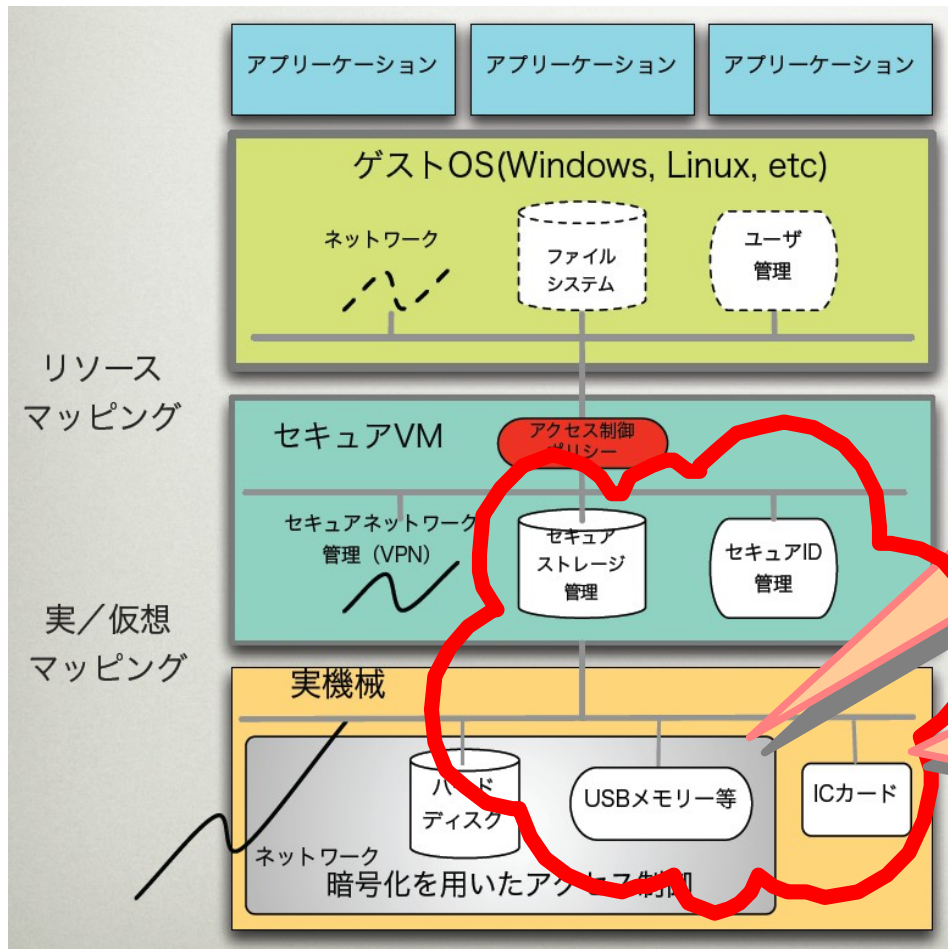
松原 克弥  
株式会社イーゲル

2008.11.18

# 本発表の目的

- BitVisorでUSBを使うときの動きを理解する。
- BitVisorのUSB関連コードの読解・改良の助けとなるようにする。
- USBを仮想化する際のIssueをまとめる。

# セキュアVM(BitVisor)とUSB

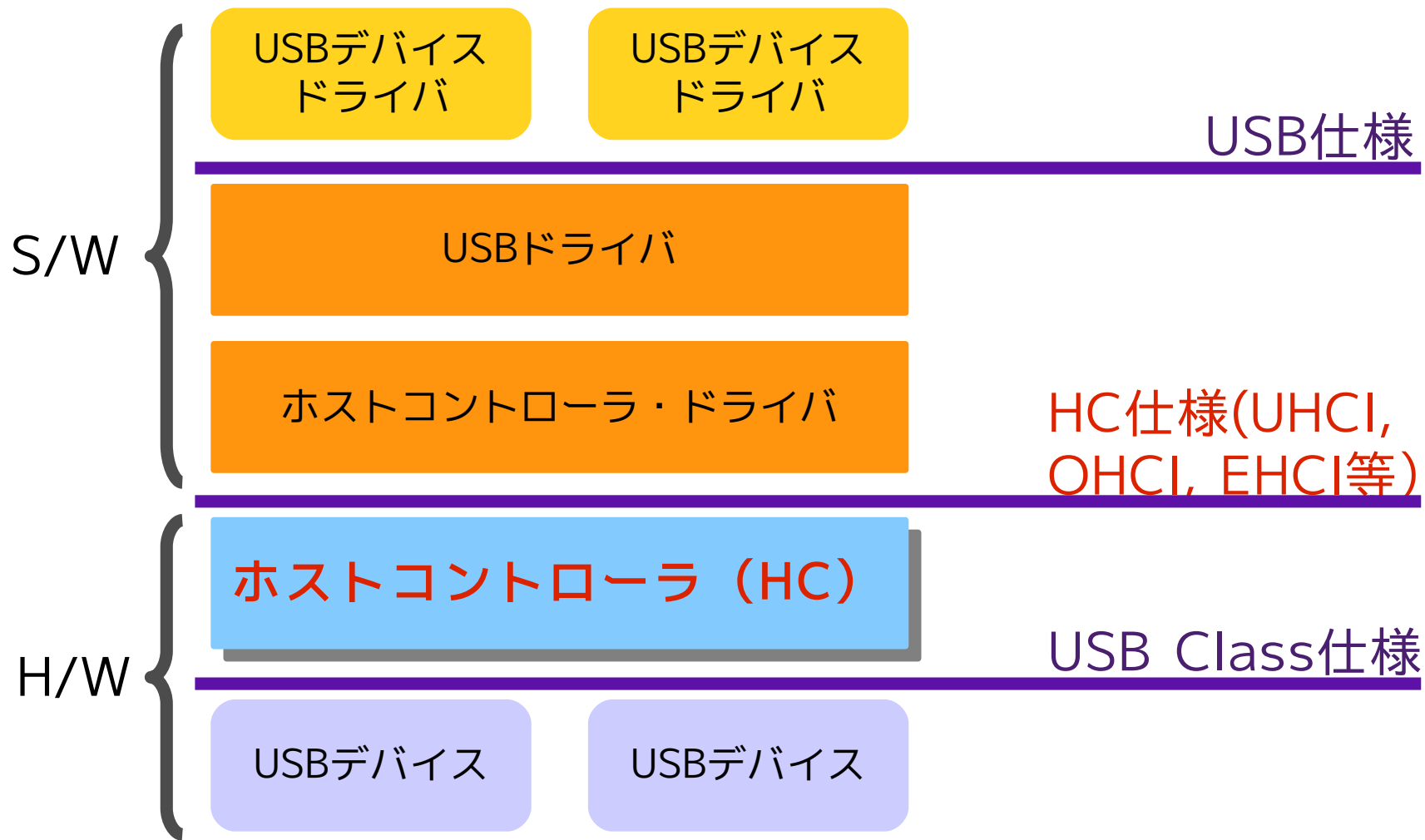


- USBストレージへの入出力データを暗号化
- 復号化

- ICカードリーダーを使ってユーザ認証や暗号キーを取得

# USBに関する仕様概要

# USB仕様：システムブロック



# USB仕様：転送種別

## ■ Isochronous転送

帯域保証があるが、転送保証のない（エラーの場合に再送しない）転送種別。マルチメディア向け

## ■ Interrupt転送

帯域保証と転送保証のある転送種別で少量の入力データ用。キーボードやマウス向け。

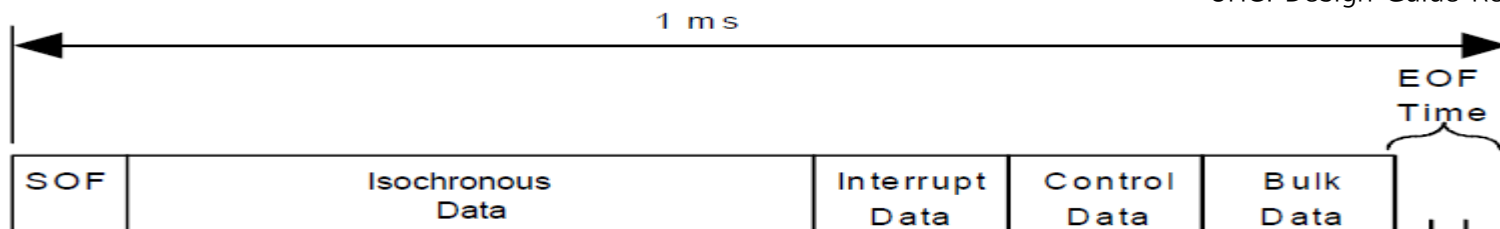
## ■ Control転送

転送保証のある転送種別で、デバイス制御向け。

## ■ Bulk転送

帯域保証はないが、転送保証のある（再送あり）転送種別。大容量データ転送向け

UHCI Design Guide Rev.1.1より抜粋



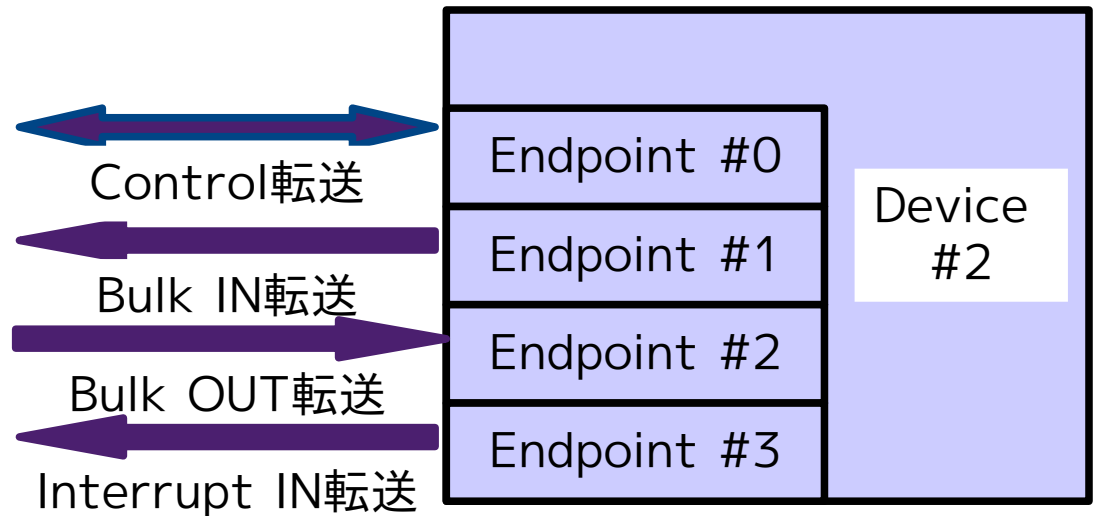
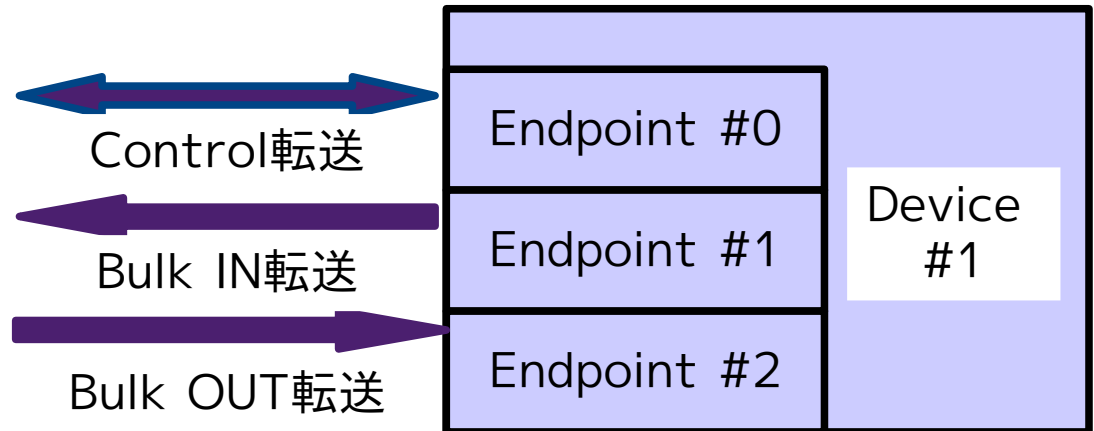
# USB仕様：転送ポート

## ■ デバイスアドレス

- 0x01から0x7fの値
- HCドライバにより管理
- HC毎に一意

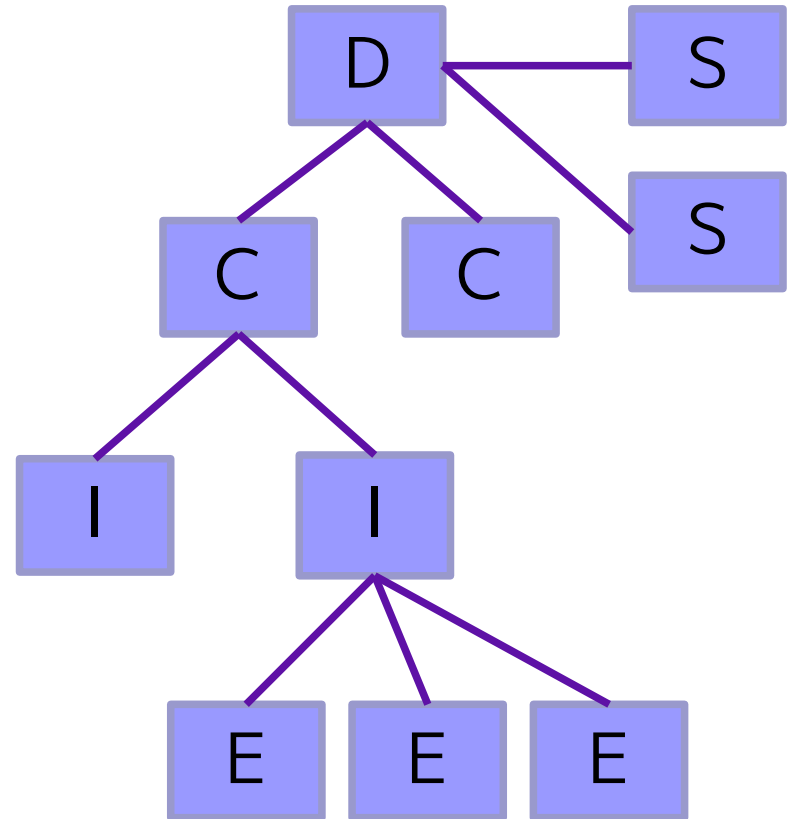
## ■ エンドポイント

- 転送種別と方向別
- Control転送用の「#0」は必須



# USB仕様：ディスクリプタ

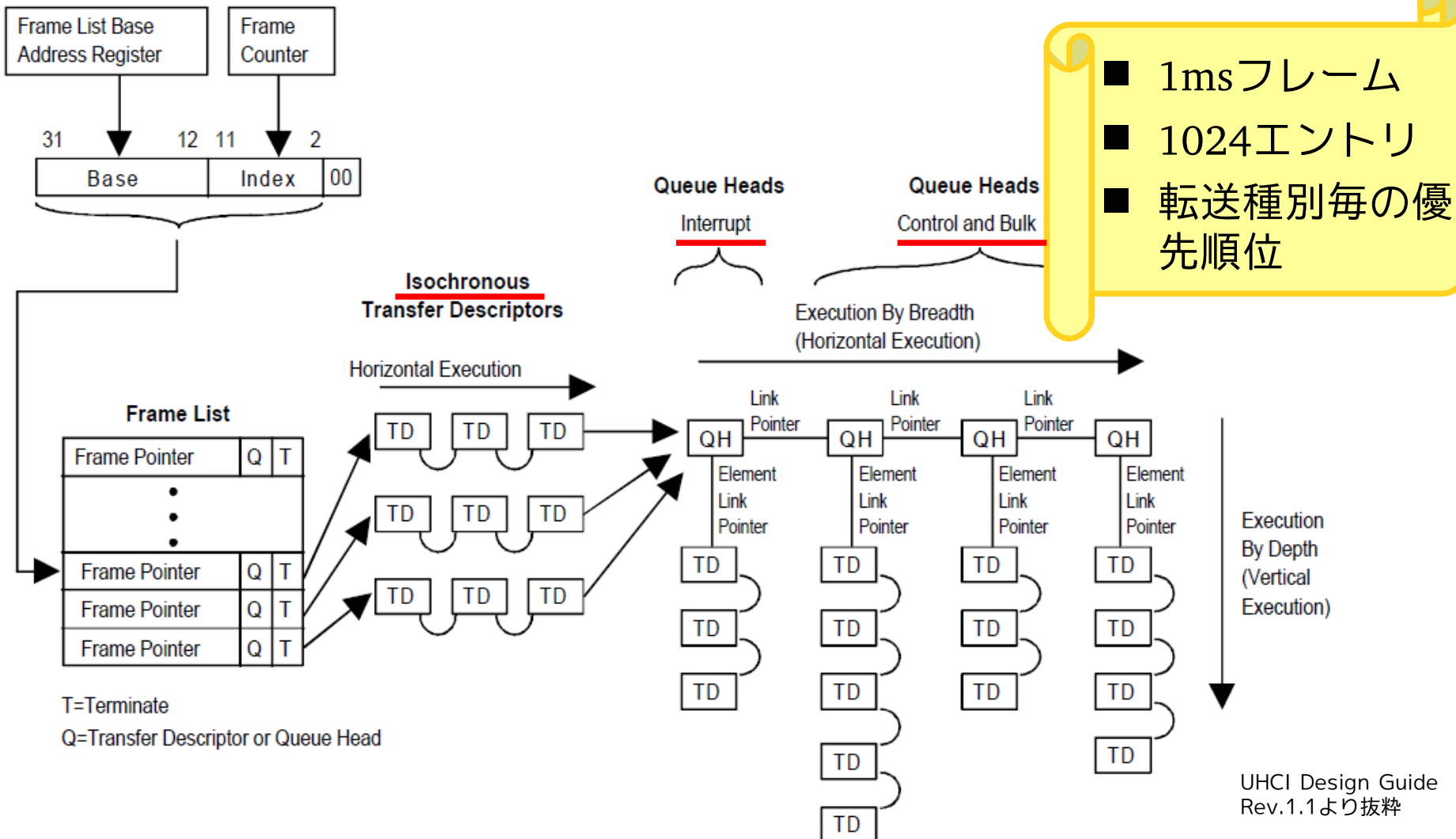
- Device descriptor  
デバイスに関する情報
- Configuration descriptor  
後述構成に関する情報
- Interface descriptor  
インタフェースに関する情報
- Endpoint descriptor  
エンドポイントに関する情報
- String descriptor  
文字列に関する情報



# UHCI仕様：レジスタ

- USBCMD  
HCの開始／停止／リセットを制御
- USBSTS  
HCおよびTransferデータ（Transaction）の状態取得
- PORTSC  
ポートの状態取得および制御
- FRBASEADD  
フレームリストのベースアドレス設定

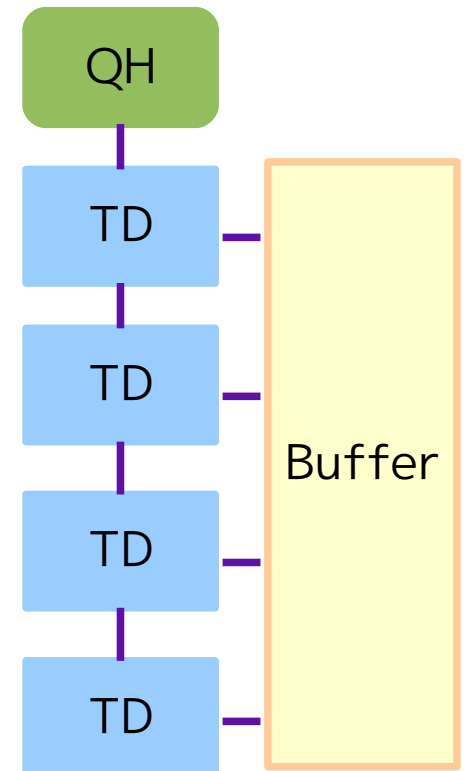
# UHCI仕様：フレームリスト



- 1msフレーム
- 1024エントリ
- 転送種別毎の優先順位

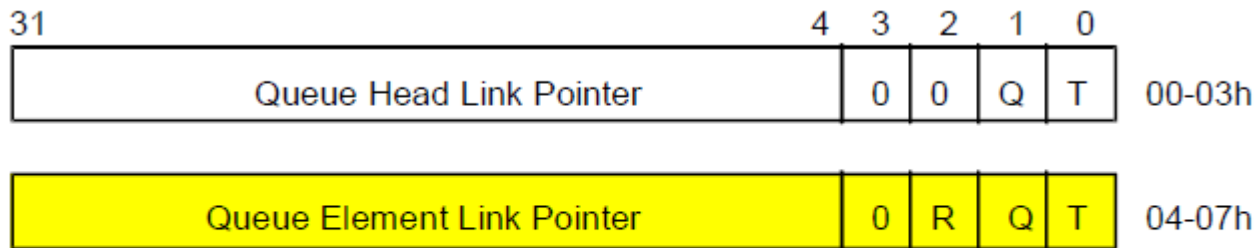
# UHCI仕様：Transfer Queue

- Isochronous転送以外の転送データ形式
- 1つのQueue Head(QH)と複数のTransaction Descriptor(TD)から構成
- TDはパケット毎

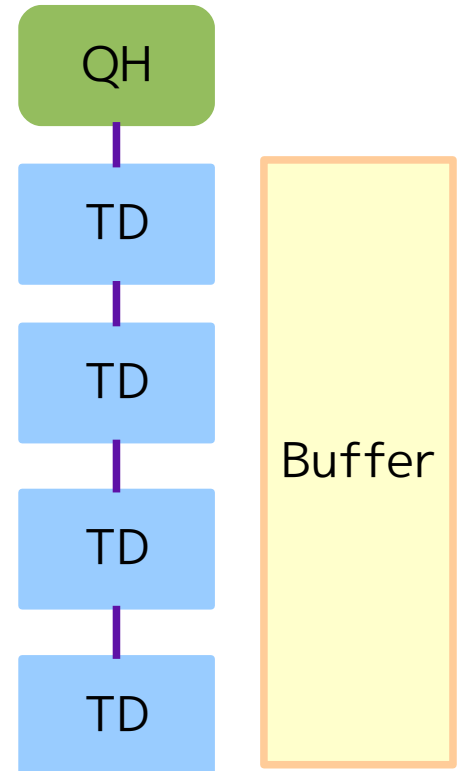


# UHCI仕様：Queue Head (QH)

- Link  
次のQHへのポインタ
- Element  
TDへのポインタ



Host Controller Read/Write     Host Controller Read Only

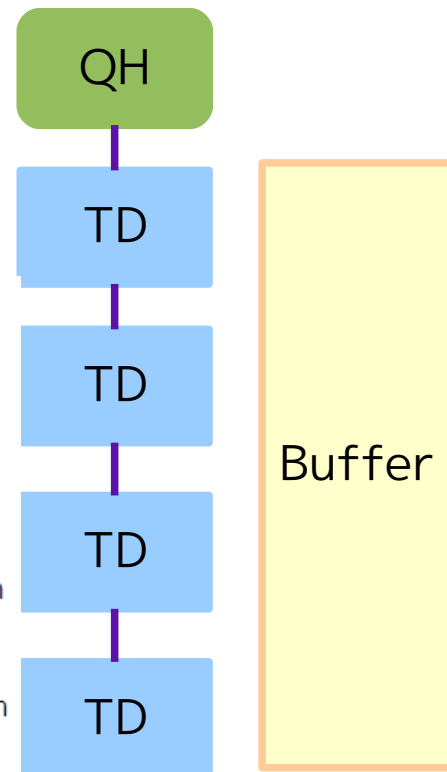
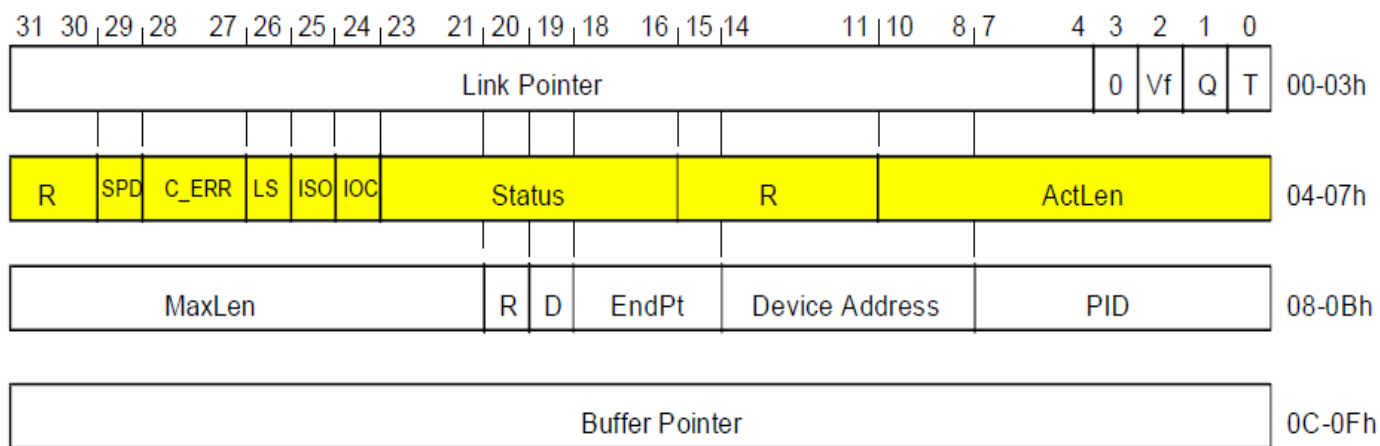


UHCI Design Guide Rev.1.1より抜粋

# UHCI仕様 : Transaction Descriptor (TD)



- Link
- Status
  - Active
  - Interrupt on Complete (IOC)
- Token
  - EndPt
  - Device Address
- Buffer



R=Reserved

UHCI Design Guide Rev.1.1より抜粋

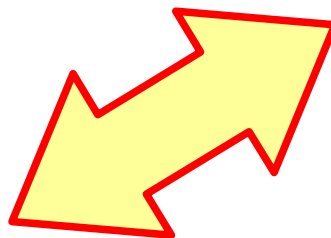
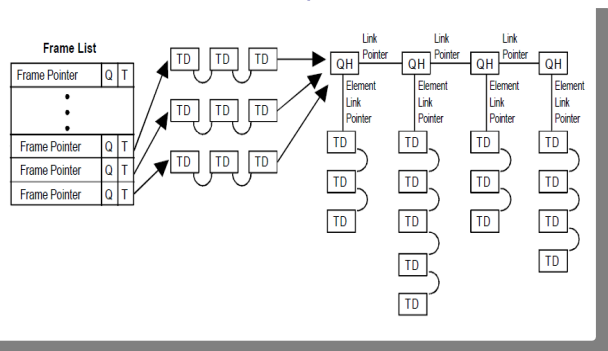
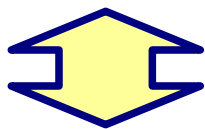
■ Host Controller Read/Write    □ Host Controller Read Only

# 実装 UHCIの仮想化

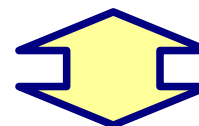
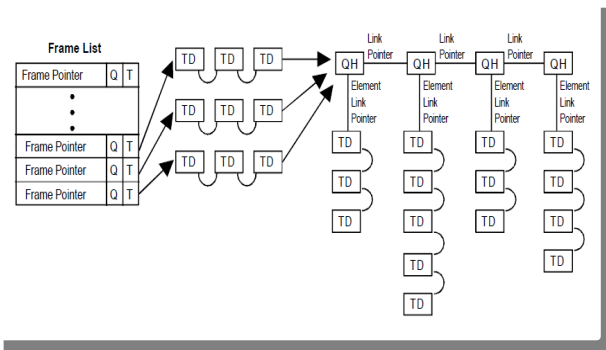
目的：ゲストOSとHC間のTransfer Queueを監視・加工

- フレームリストのシャドウ化
- レジスタアクセスはFRBASEADD以外を基本的にパススルー

ゲストOSによる  
Transfer Queueの挿入／削除



VMによる  
Transfer Queueの  
シャドウイング



HC H/Wによる  
Transfer Queueの処理

## ■ ゲストOSの実装仕様に関連する課題

1. Transfer Queue制御方式
2. タイムアウト

## ■ VM機能要件に関連する課題

3. 特定パターンのTransferデータの検出
  - ・ USBメモリへの入出力データの暗号化／復号化
4. VM内からのTransfer Queue作成
  - ・ ICカードリーダーへの入出力

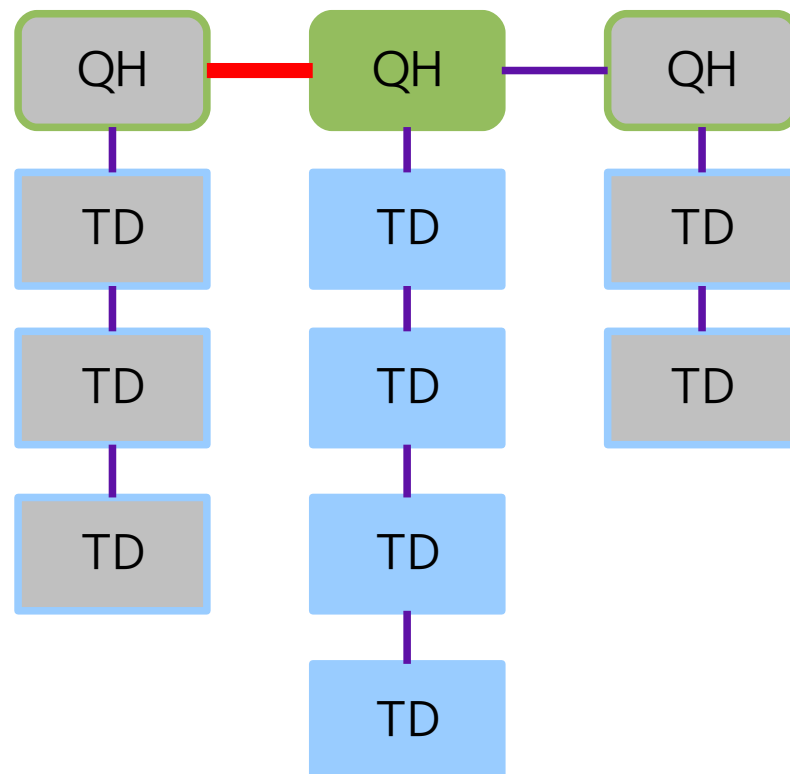
# 1. Transfer Queue制御方式： Transfer Queueの連結手順

- 新規Transfer Queueの投入手順に、レジスタへのアクセスを行わない。
  - HCは動作させたまま、フレームリスト内のQHに対するポイント更新のみでTransfer Queueを追加投入
  - 任意メモリ領域へのアクセスを検出できないBitVisorでは、Transfer Queueの投入タイミングを検出できない。
- Transfer Queueの投入パターンが様々（仕様外）
  - 本開発では、4パターンを発見

# Transfer Queue投入パターン 1

## 新しいQH+TDsを連結

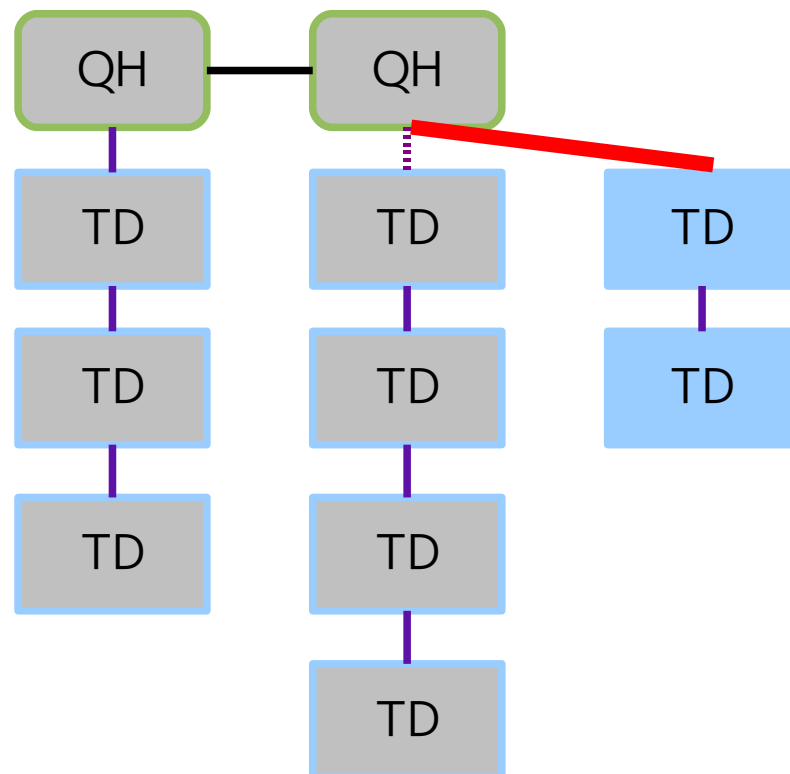
→ 多くのOSで発見



# Transfer Queue投入パターン 2

既存QHを再利用して、新しいTDsを連結

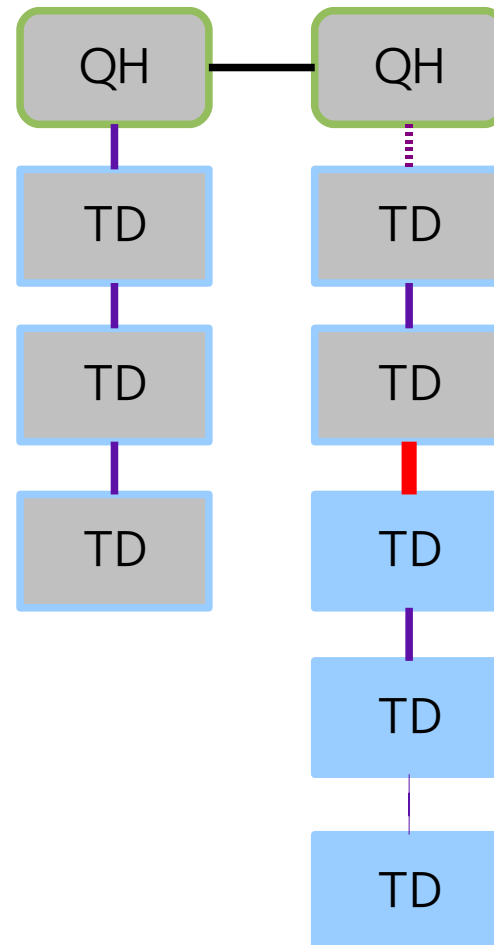
→ 一部のゲストOSで発見



# Transfer Queue投入パターン 3

## 既存QH+TDsの末尾に新しいTDsを連結

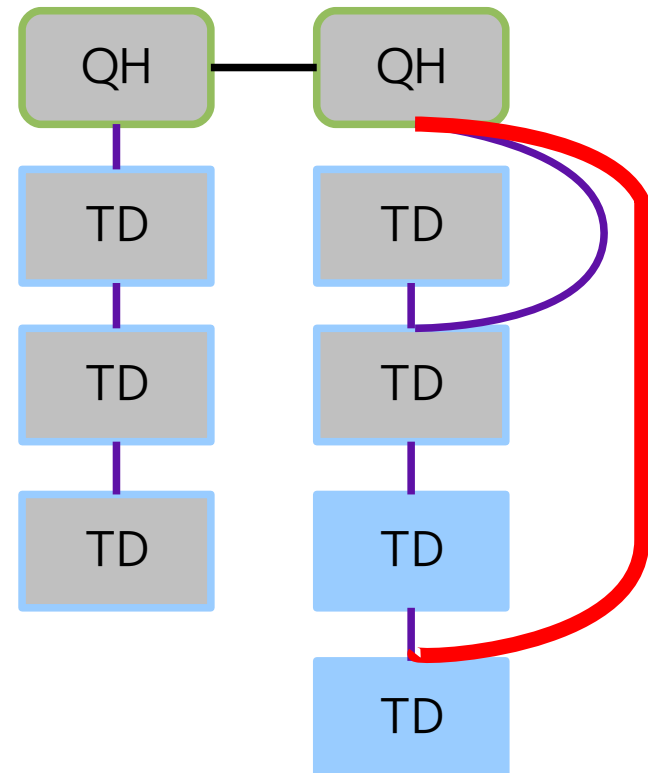
- Linuxにおいて発見
- 同じエンドポイント宛のTransfer Queueが存在する場合に利用



# Transfer Queue投入パターン 4

## 未処理のTDへポインタ更新

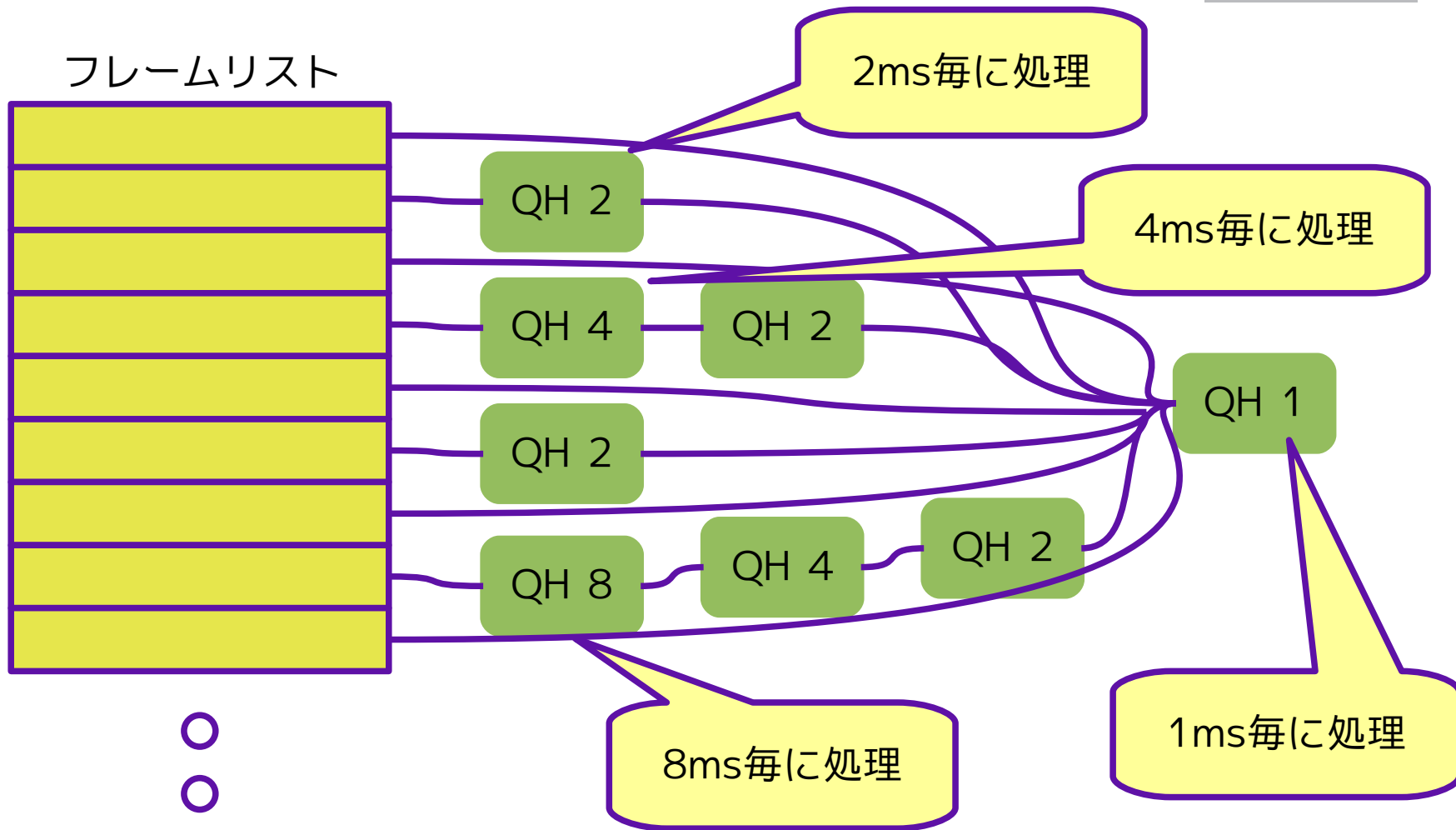
- Control転送において利用
- Short Packet（期待した転送量よりも少ない結果）が発生したときに利用



# 1. Transfer Queue制御方式：スケルトンQH

- フレームリストの初期化時に、ダミーのQHを配置
  - 1msを越えるIntervalのInterrupt転送の挿入位置が容易に
  - 明示的な割り込み発生を実現
- スケルトンQHの数はゲストOS実装依存
  - 8個 (Linux 2.6.23)
  - 11個 (Linux 2.6.18)
  - 65個 (Windows XP/Vista)

# スケルトンQHの構成



# 実装：スケルトンQHのシャドウ イング

- ゲストOSのフレームリスト構成に依存せず、一定（8個）のスケルトンQHを作成
- ゲストOSのスケルトンQHの出現回数をカウントして、対応するInterval値を計算
- シャドウTransfer Queueの投入時は、スケルトンQHから計算したInterval値を考慮して挿入位置を決定

# 実装：フレームリストモニタ

- VMが動作する毎にフレームリスト（QH）をスキャン
- 未知のTransfer Queueを発見＝新規Transfer Queueが追加された。
- 既知のTransfer Queueが見つからない＝Transfer Queueが削除された。
- スケルトンを含むフレームリスト内の以下のエントリの値を保持し、変更を検出
  - QHのLink Pointer（パターン1に対応）
  - QHのElement Pointer（パターン2、4に対応）
  - 末尾TDのLink Pointer（パターン3に対応）を監視

## 2. タイムアウト

- VMが動作する間隔が大きい。
  - 典型的な原因：Tick-less kernel
- 仕様に時間の概念（1msフレーム）があることから、ゲストOSが設定するタイムアウト値が小さい。
  - Linux UHCIドライバでは200ms

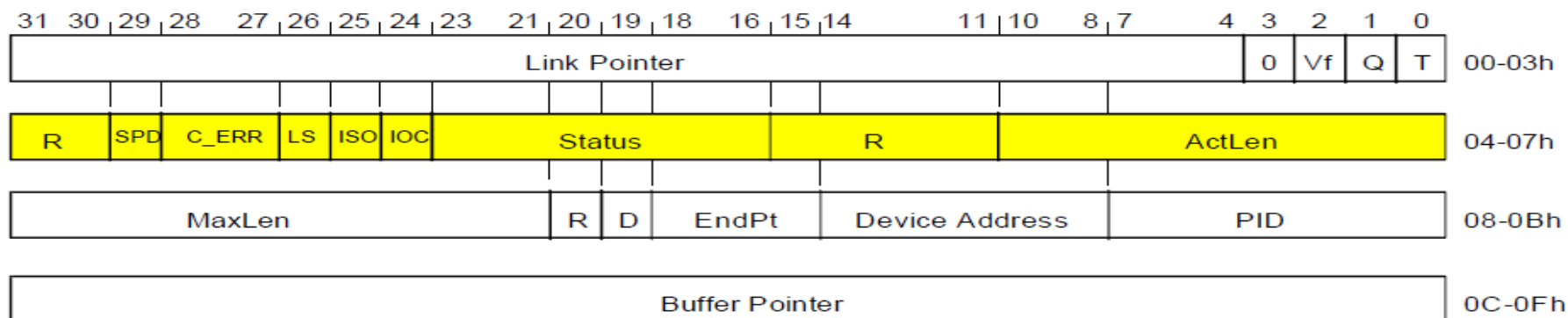
# 実装：フレームリスト・モニタ のブースト



## TDのIOCビットを使って、一定間隔で割り込みを発生

- 割り込み間隔は、ダミーTDの連結対象となるスケルトンTDのInterval値に対応

24	<p><b>Interrupt on Complete (IOC).</b> 1=Issue IOC. This bit specifies that the Host Controller should issue an interrupt on completion of the frame in which this Transfer Descriptor is executed. Even if the Active bit in the TD is already cleared when the TD is fetched (no transaction will occur on USB), an IOC interrupt is generated at the end of the frame.</p>
----	---



R=Reserved

Host Controller Read/Write  
 Host Controller Read Only

UHCI Design Guide Rev.1.1より抜粋

# 4. 特定入出力データの検出

## ゲストOS Transaction (TDとバッファデータ) に対する パターンマッチ機能を実装

- デバイス情報（アドレス，ディスクリプタ）の取得に利用
- ストレージデータ入出力の暗号化／復号化を実現

# 5. VM内からのデータ入出力

## libusb互換のAPIを実装

- シャドウフレームリストにTransfer Queueを直接投入
- libusbを使って書かれたデバイスドライバ実装を流用できる。

- コード量は約6500行
- Linux 2.6.18/2.6.23, Windows XP/VistaをゲストOSとして動作を確認
- USBメモリの暗号化／復号化を実現
- ICカードリーダーに対する入出力を実現
- BitVisor 0.7にマージ予定

## UHCI HCの仮想化

- フレームリストをシャドウ化
- ゲストOSのフレームリストを監視することでTransfer Queueを抽出、シャドウイング
- IOC割り込みを使って監視頻度を調整
- TDおよびバッファデータのパターンマッチによるフック機能を実現
- libusb互換のAPIを実装

# 今後の課題

- ポート管理
  - デバイスの切断対応
  - Issue: USBハブとルートハブ(PORTSC) の違い
- デバイスの偽装（隠蔽）
  - ゲストOSに対してデバイスの存在を隠蔽
  - Issue: デバイスアドレス付与
- シャドウ方式の比較
  - メモリアクセス検出とポーリング（モニタ）
  - Issue: 不必要なアクセスの検出
- EHCIへの対応
  - 現状、デバイスの多くはUSB 2.0対応
  - Issue: タイムアウト
- 性能評価