UEFI & EDK II Base Training

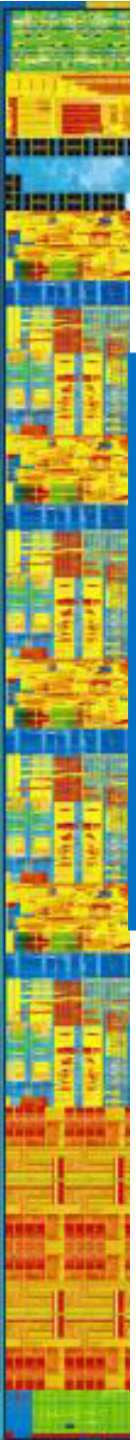
UEFI Human Interface Infrastructure

Intel Corporation
Software and Services Group

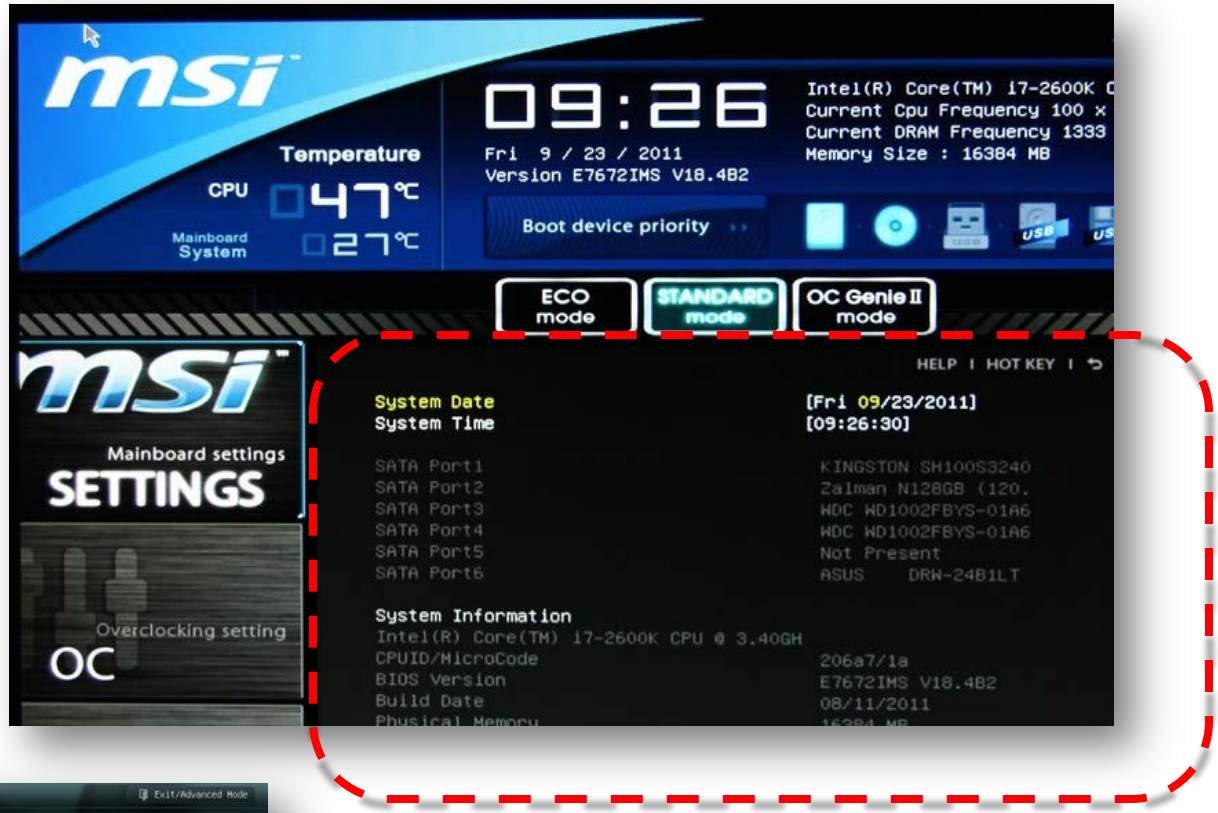
OBJECTIVE:

- What is the Infrastructure for HII
- How Does HII work
- Lab

User Interface HII Overview



WHY ?



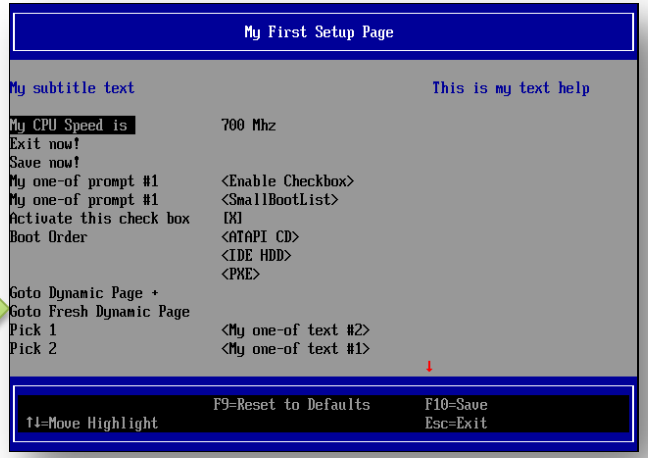
- Unified Look and Feel at Platform level
- Single Interface
- Localization

HII: Key Concepts

localization

forms & strings

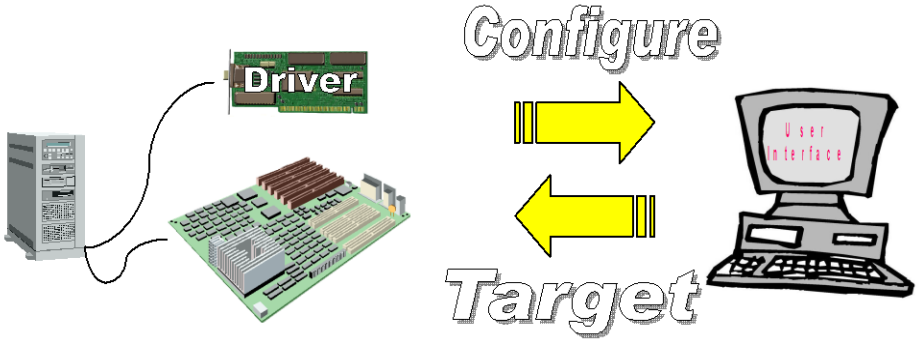
HII

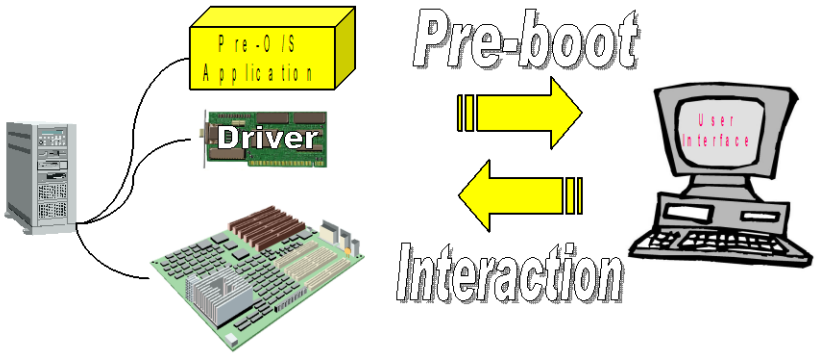


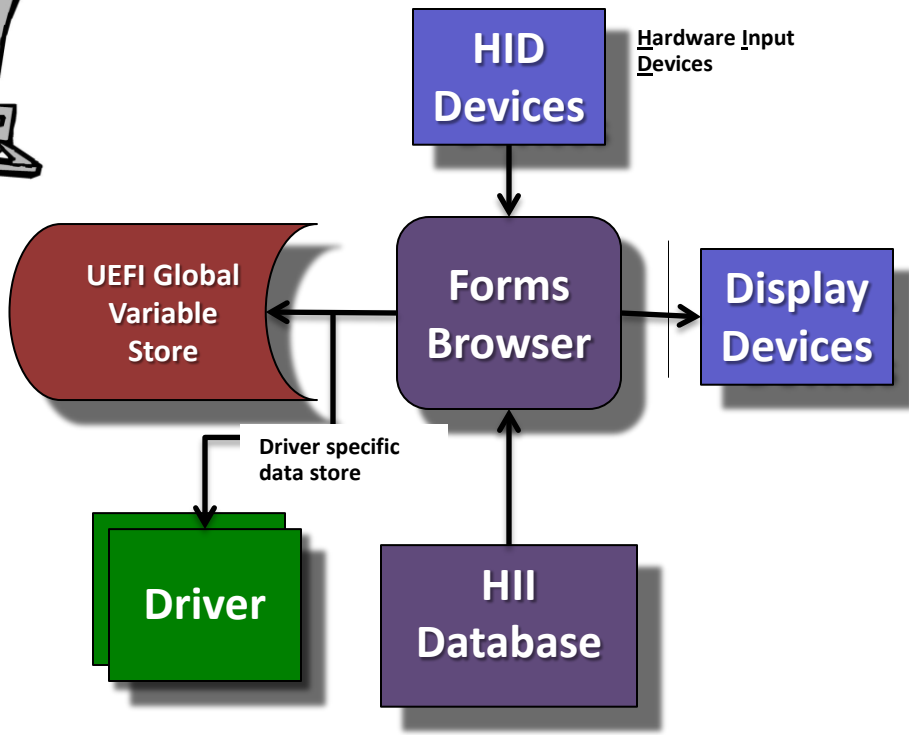
setup browser

input sources

Design Discussions

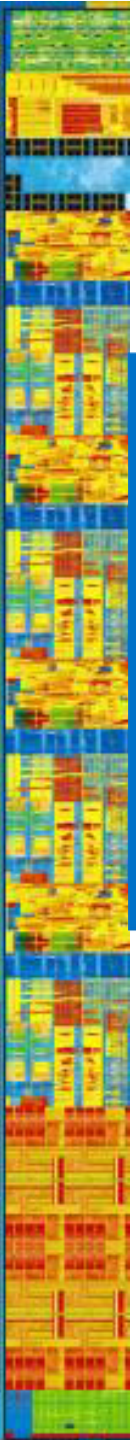






See § 28.2 of the UEFI 2.3x Spec.

HII Components



Human Interface Components

Strings

Fonts

Keyboard

Forms

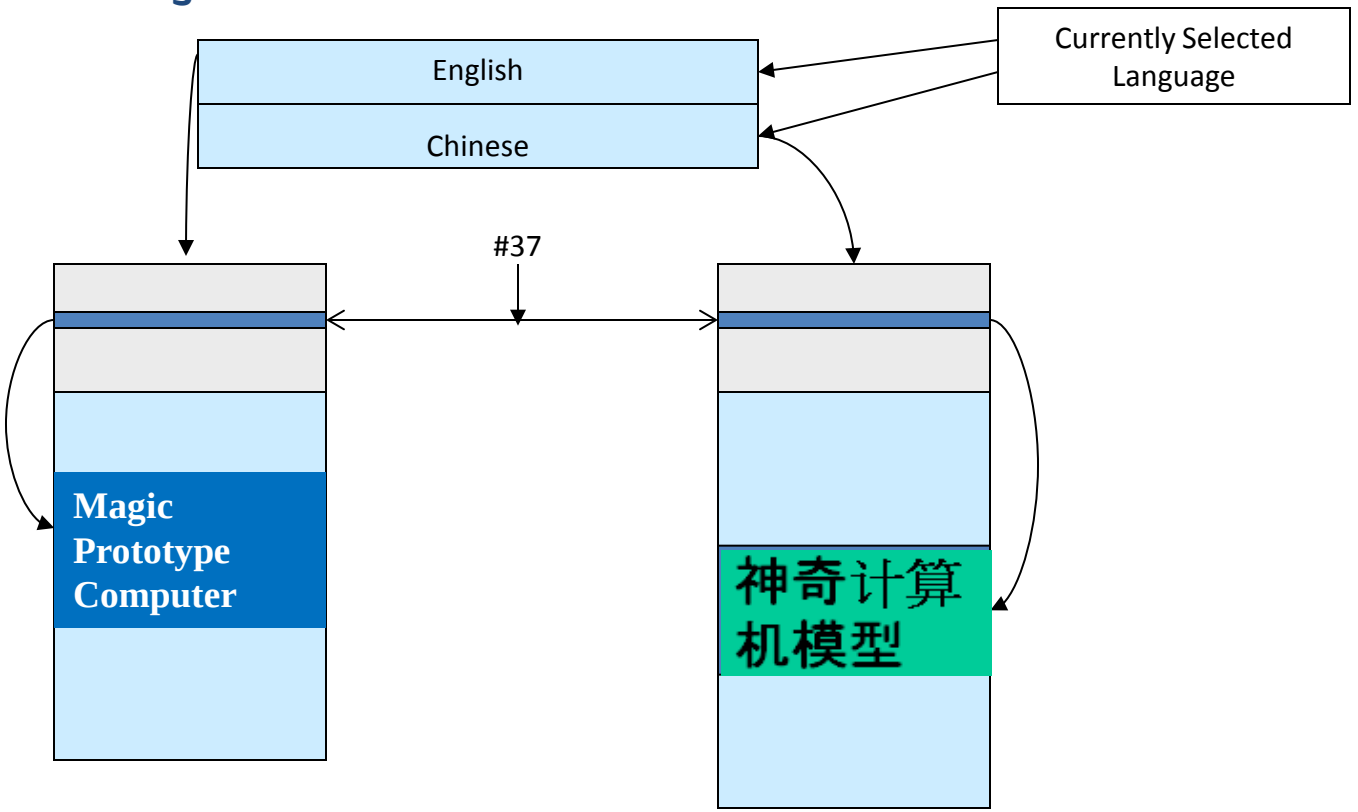
Packages

Strings

- Strings stored in Unicode
  - Real string encodings required for e.g. VT100
  - Already the text standard in UEFI today
- Localization happens at the string level
  - Caller externs and passes in language independent string token
  - String support determines actual string from token and selected language
  - Usage Model:
    - A string library supporting translations
      - Reduces translation costs and delays
    - Tools to extract strings depending on use by driver
    - Analysis of strings used to extract fonts
    - RFC 4646 Language codes (2-2)

Token to String Mapping

- Request: Print string with token 37
- Currently selected language is as in UEFI 2.X. This is used to select between language data structures. (The structures indicate which language(s) they support).
- The top part of the structure maps from token to string. The bottom part of the structure is the strings



String Example (.uni file)

```
#langdef   en-US      "English"
#langdef   fr-FR      "Francais“
#langdef   sv-SE      “Svenska”

#string STR_FORM_SET_TITLE          #language en-US "Browser Testcase Engine"
                                    #language fr-FR "Navigateur Testcase Moteur”
                                    #language sv-SE "Webbläsare Testcase Motor”

#string STR_FORM_SET_TITLE_HELP     #language en-US "This is a sample UEFI driver which is used
                                        to test the browser op-code operations. "
                                    #language fr-FR “Il s'agit d'une UEFI Driver échantillon qui
                                        est utilisé pour tester les navigateurs op-code
                                        opérations.“
                                    #language sv-SE ”Detta är ett exempel på UEFI-drivrutin som
                                        används för att testa webbläsaren op-kod operationer”

#string STR_FORM1_TITLE             #language en-US "My First Setup Page"
                                    #language fr-FR "Mi Primero Arreglo Página“
                                    #language sv-SE "Min första inställningssidan”
```

RFC 4646 Language codes

Fonts

- One Standard Font for UEFI
  - One font database accumulated during boot
- Each Component Provides Its Fonts
  - System provides ASCII and ISO Latin-1
  - Fonts only required for characters in strings that may appear
    - If the firmware will never print “tractor” in Kanji, discard the bit image
  - Result is a sparse array of characters indexed by the Unicode ‘weight’
- Wide and Narrow glyphs supported

Keyboard

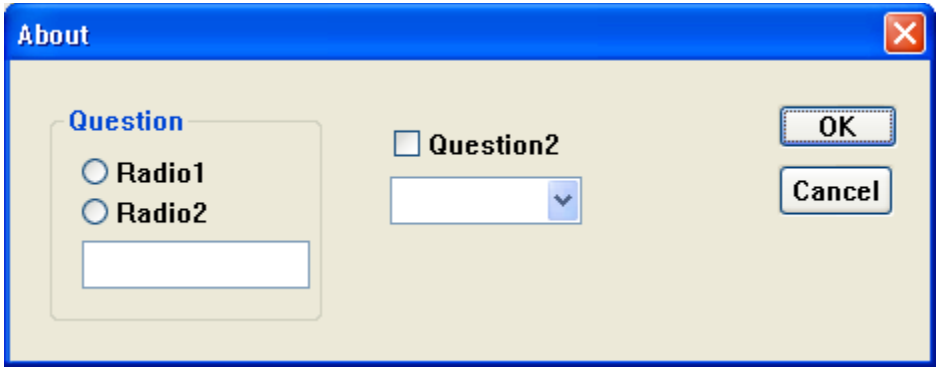
- Support varying keyboards
  - UK and US keyboard layout are not the same. Certainly that is the case for US and Arabic, etc.
  - Adding support of other modifiers (e.g. Alt-GR, Dead-keys, etc)
- Keyboard Layout
  - Allow for a standardized mechanism to describe a keyboard layout and add to system database.
  - Allow for switching keyboard layouts.

Spanish

English

French

Forms



- The forms are stored in the HII database, along with the strings, fonts and images
- Other applications may use the information within the forms to validate configuration setting values
- The Forms Browser provides a forms-based user interface which understands
  - how to read the contents of the forms
  - interact with the user
  - save the resulting values
- The Forms Browser uses forms data installed by an application or driver during initialization in the HII database.

Visual Forms Representation (VFR)

- Language used to describe what a page layout would be in a browser as well as the op-codes and string tokens to display
- Op-codes are defined for the following functions examples
  - `formSet` and `form` definitions
  - One of type questions with corresponding options (combo) fields
    - `checkbox`
    - `numeric`
    - `oneof`
    - `String`
  - Boolean expressions in support of errors, suppress, and gray outs
    - `"disableif"`
    - `“suppressif“`
    - `“grayoutif"`

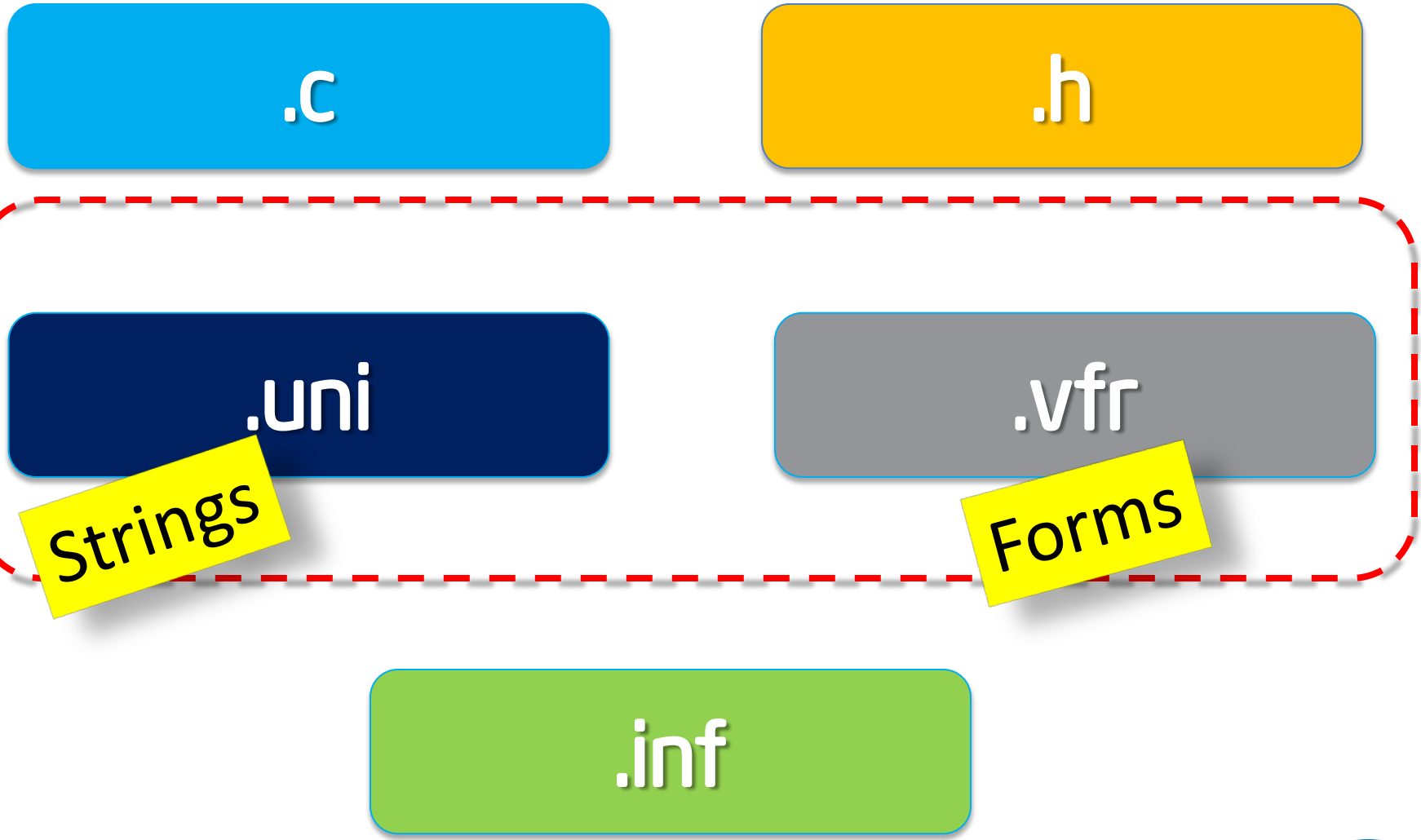
Form Example (.vfr file)

```
formset
  guid     = FORMSET_GUID,
  title    = STRING_TOKEN(STR_FORM_SET_TITLE),
  help     = STRING_TOKEN(STR_FORM_SET_TITLE_HELP),
  classguid = EFI_HII_PLATFORM_SETUP_FORMSET_GUID,

 varstore   DRIVER_SAMPLE_CONFIGURATION,
    name = MyIfrNVData,
    guid = FORMSET_GUID;

 form formid = 1,
       title  = STRING_TOKEN(STR_FORM1_TITLE);

 oneof varid  = MyIfrNVData.MyVariableForOneofPrompt,
  prompt      = STRING_TOKEN(STR_ONE_OF_PROMPT),
   help       = STRING_TOKEN(STR_ONE_OF_HELP),
  option text = STRING_TOKEN(STR_ONE_OF_TEXT1), value = 0x0, flags = 0;
  option text = STRING_TOKEN(STR_ONE_OF_TEXT2), value = 0x1, flags = 0;
  option text = STRING_TOKEN(STR_ONE_OF_TEXT3), value = 0x2, flags = DEFAULT;
 endoneof;

             • • •

 endform;
endformset;
```

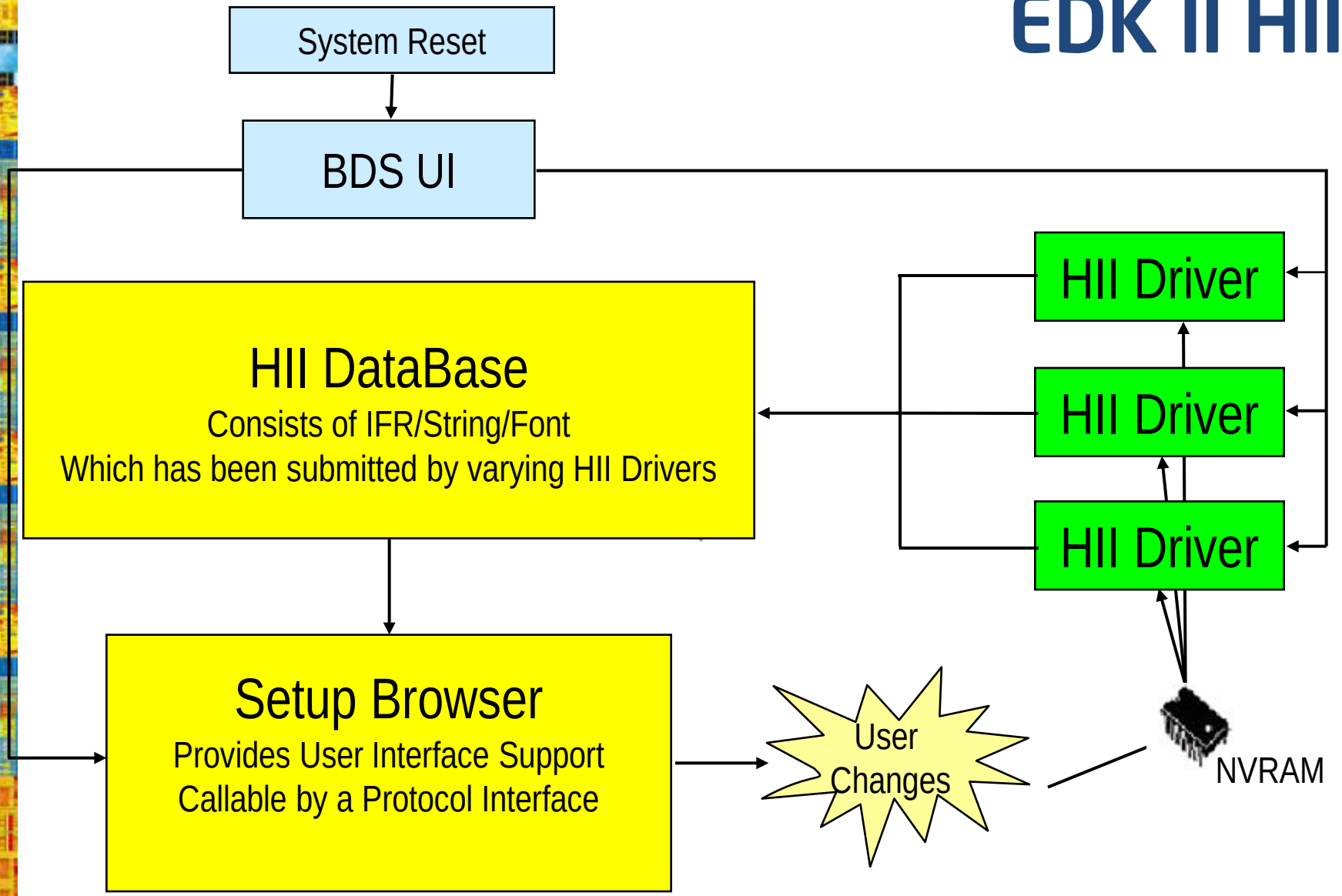
Internal Forms Representation (IFR)

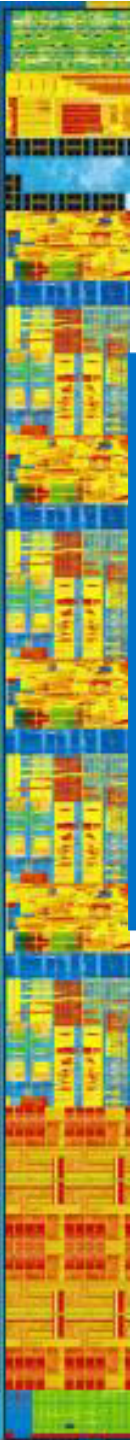
- IFR Code created by VFR to IFR compiler tool
- Byte encoded operations (much smaller)
- String references abstracted as tokens
- Improved validation, visibility primitives
- At better level of presentation control for firmware
  - Tension between configuration driver and presentation driver over control of presentation format
- Easy to
  - Interpret for small Setup engine in desktop firmware
  - Translate into XHTML or JavaScript or …

Minimum Files for HII Driver



EDK II HII

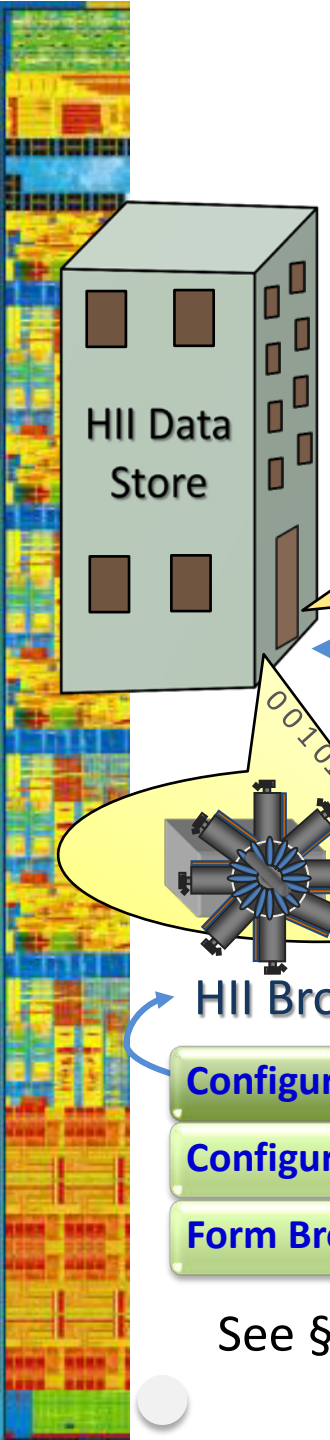

System Reset
BDS UI
HII DataBase
Consists of IFR/String/Font
Which has been submitted by varying HII Drivers
HII Driver
HII Driver
HII Driver
Setup Browser
Provides User Interface Support
Callable by a Protocol Interface
User Changes
NVRAM



How: UEFI HII Protocols

Sections 28-30 the UEFI 2.3x Specification

HII Database Overview



HII Data Store

Data

- Fonts, Strings, Image, Forms
- GUID, Keyboard Layout, Device Paths

NVRAM

HII Protocols

- Font Protocol
- String Protocol
- Image Protocol
- Database Protocol

See § 29 of the UEFI 2.3x Spec.

HII Browser Engine Protocols

- Configuration Routing Protocol
- Configuration Access Protocol
- Form Browser 2 Protocol

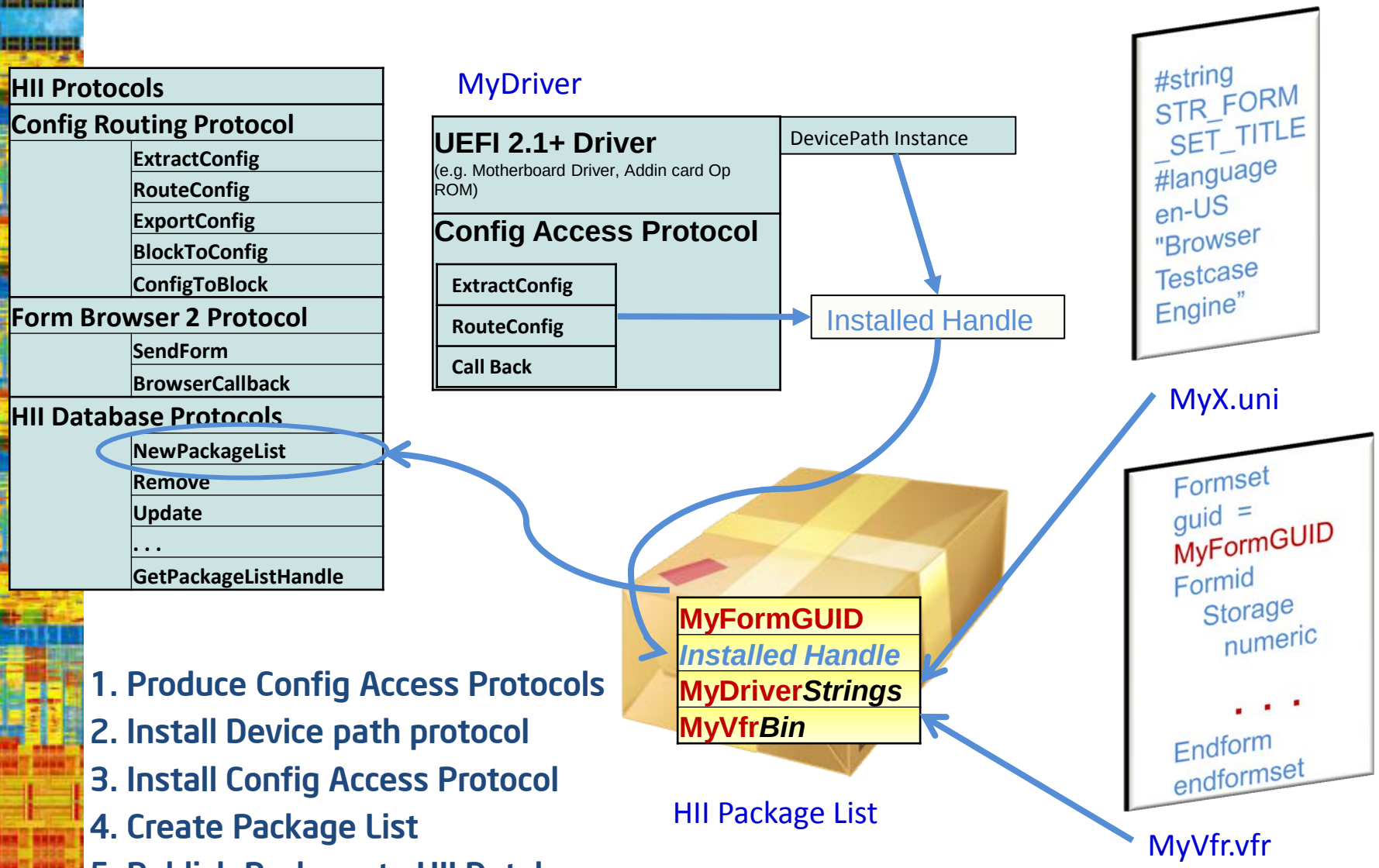
See § 30 of the UEFI 2.3x Spec.

UEFI HII Protocols

Font Protocol	• Sting to Image, Sting ID to Image, Get Glyph, Get Font Info
String Protocol	• New – Get – Set – String • Get Language & 2nd Language
Image Protocol	• New – Get – Set Image • Draw Image, Draw Image ID
Database Protocol	• New – Remove- Update – List – Export Lists – Get Handle Package • Register, Unregister Package Notify • Find- Get- Set Keyboard layout

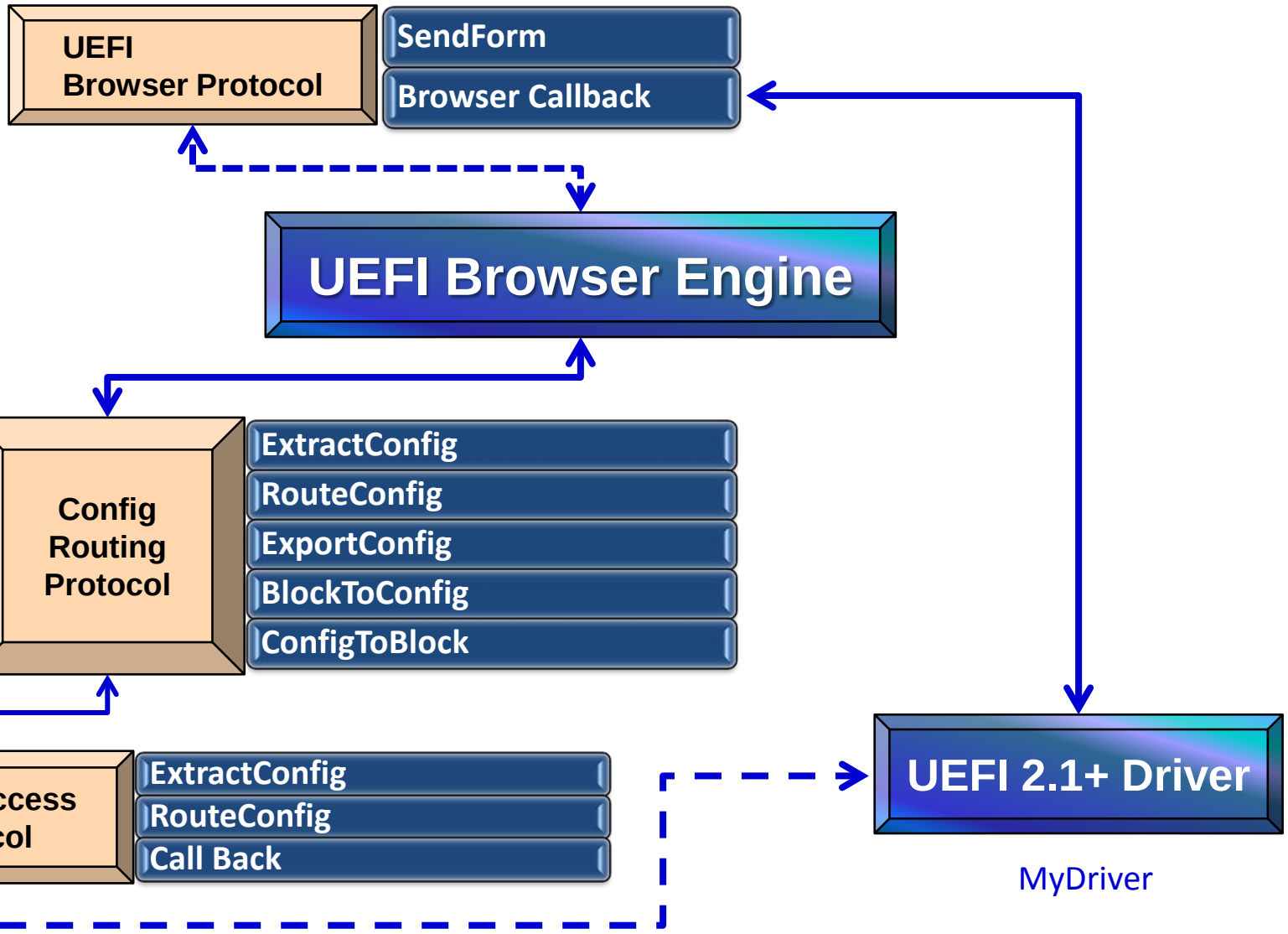
See § 29 of the UEFI 2.3x Spec.

UEFI Driver Initialization Process

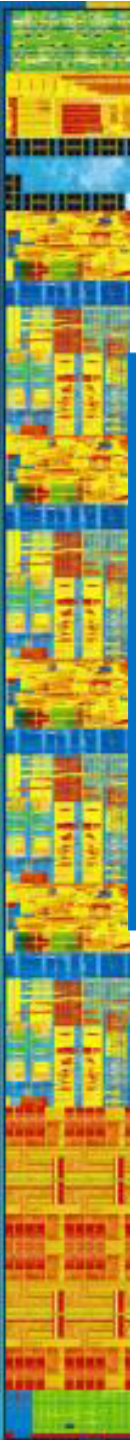


1. Produce Config Access Protocols
2. Install Device path protocol
3. Install Config Access Protocol
4. Create Package List
5. Publish Package to HII Database

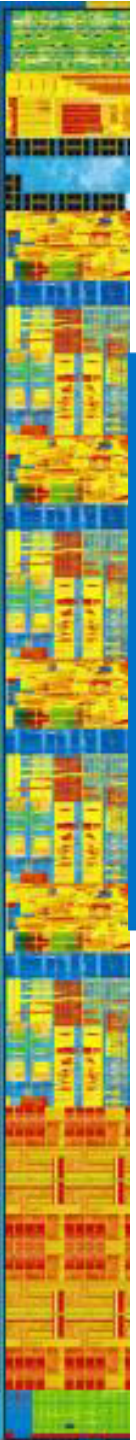
Form Browser Protocols



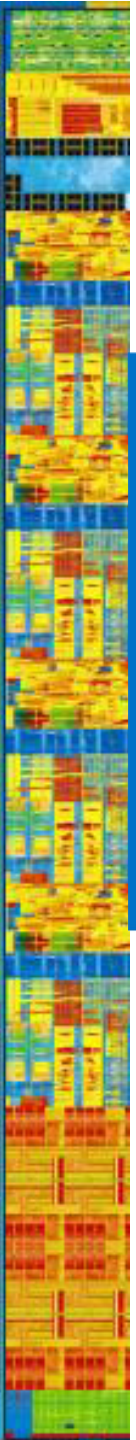
Lab 9.1 Adding HII to a UEFI Driver from the UEFI Driver Wizard



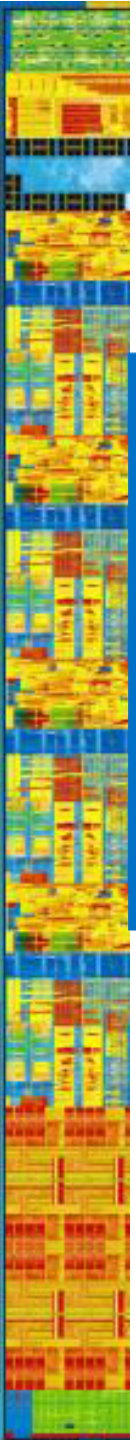
Lab 9.2 Updating HII to Save Data Settings



Lab 9.3 Updating your driver to Initialize data from the VFR data to the HII Database

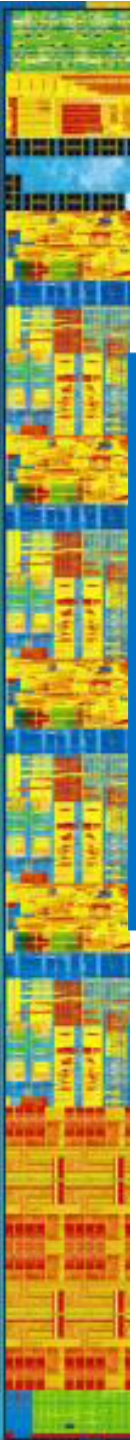


Lab 9.4 Updating Menu: Reset Button



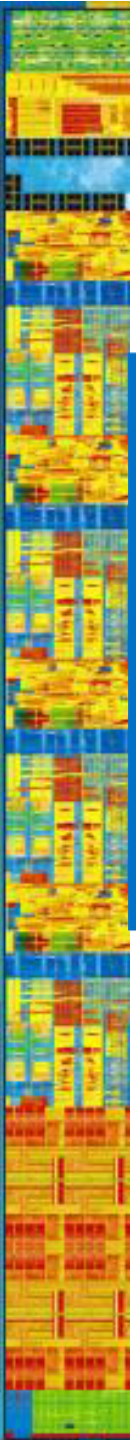
Lab 9.5 Updating Menu: Pop-up Box

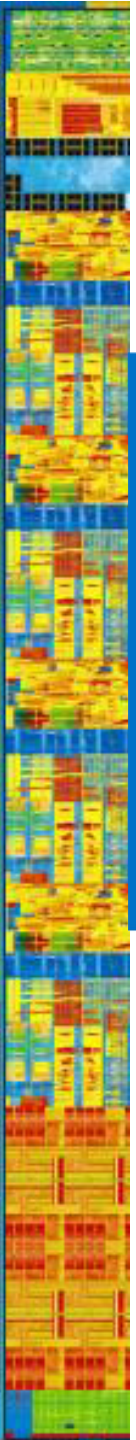
Lab 9.6 Updating Menu: Creating a string to Name the Saved Configuration



Lab 9.7 Updating Menu: Numeric Entry

Lab 9.8 Updating your Driver for Interactive Call Backs





Lab 9.9 Managing Call Back Events in Your Driver

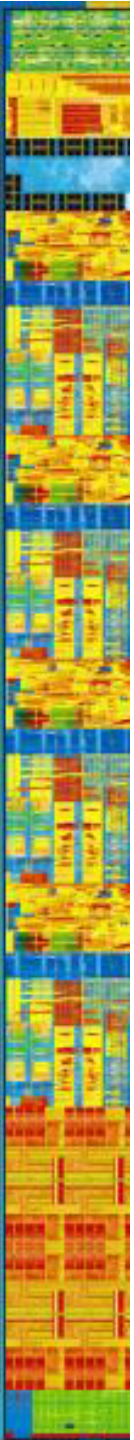
Lab 9.10 Adding Additional Form Pages

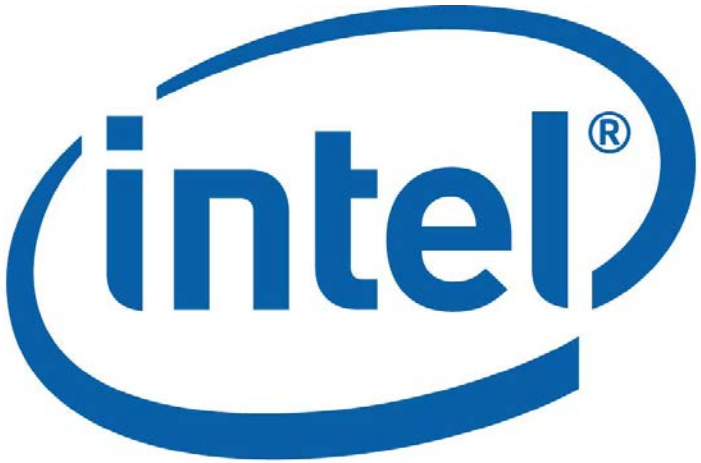
Lab 9.11 How to Add Multiple Language

Reference

- Unified Extensible Firmware Interface Specification, Version 2.3.1, http://www.uefi.org
- VFR Programming Language 1.7, http://sourceforge.net/projects/edk2/files
- Build Spec 1.22, http://sourceforge.net/projects/edk2/files


“Any questions?”





Software

BACK UP

*

Configuration Strings

Basic forms:

```
<Dec19> ::= '1' | '2' | … | '9'
<DecCh> ::= '0' | <Dec19>
<HexAf> ::= 'a' | 'b' | 'c' |  'd' | 'e' | 'f'
<Hex1f> ::= <Dec19> | <HexAf>
<HexCh> ::= <DecCh> | <HexAf>
<Number> ::= <HexCh>+
<Alpha> ::= 'a' | ... | 'z' | 'A' | … | 'Z'
```

Types

```
<Guid> ::= <HexCh>32
<LabelStart> ::= <Alpha> | "_"
<LabelBody> ::= <LabelStart> | <DecCh>
<Label> ::= <LabelStart> [<LabelBody>]*
<Char> ::= <HexCh>4
<String> ::= [<Char>]+
<AltCfgId> ::= <HexCh>4
```

Routing elements

```
<GuidHdr> ::= 'GUID='<Guid>
<NameHdr> ::= 'NAME='<String>
<PathHdr> ::= 'PATH='<UEFI binary Device Path represented as hex
number>
<DescHdr> ::= 'ALTCFG='<AltCfgId>
<ConfigHdr> ::= <GuidHdr>'&'<NameHdr>'&'<PathHdr>
<AltConfigHdr> ::= <ConfigHdr> '&'<DescHdr>
```

Configuration Strings

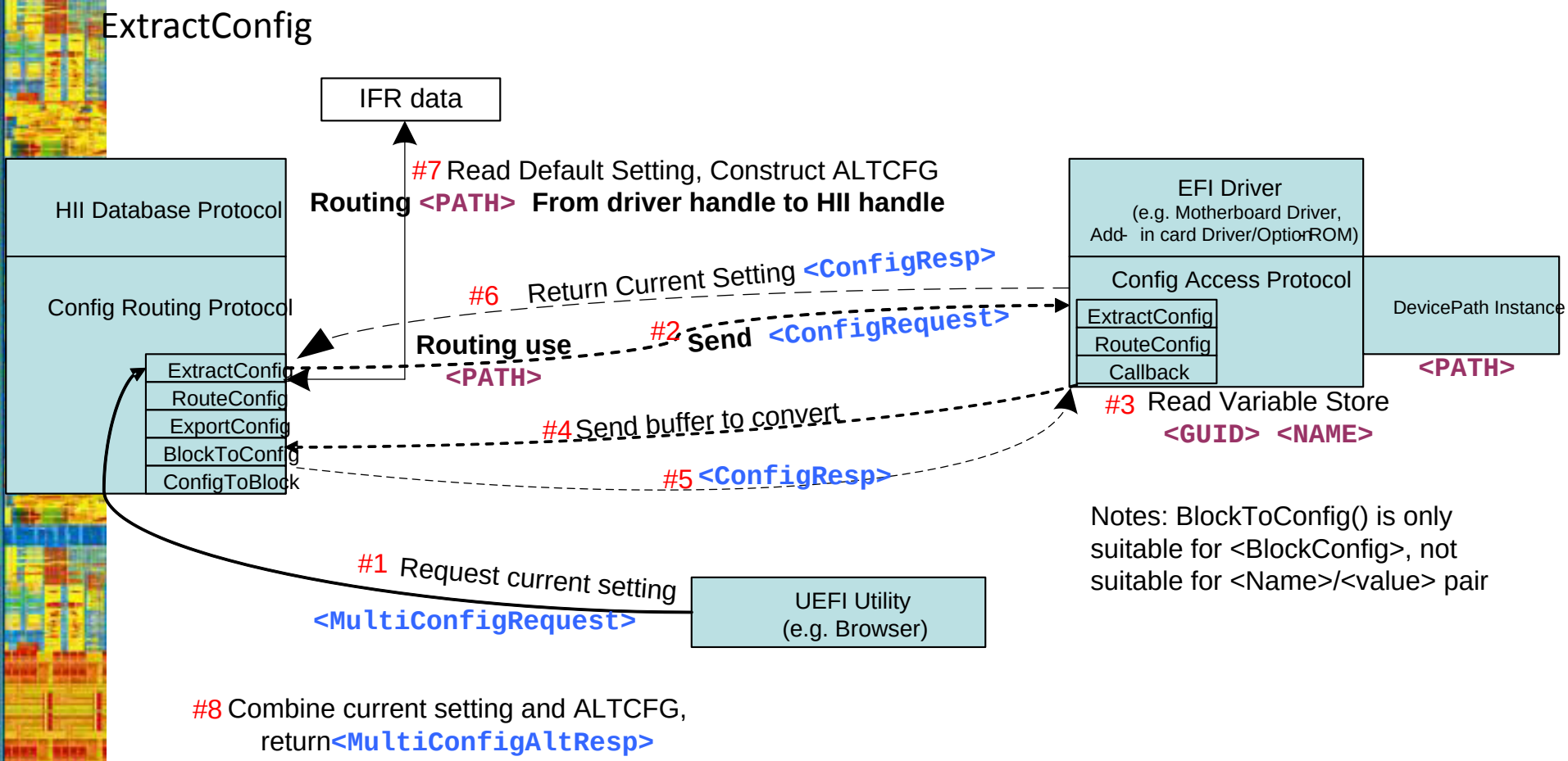
Body elements

```
<ConfigBody> ::= <ConfigElement>*
<ConfigElement> ::= '&'<BlockConfig> | '&'<NvConfig>
<BlockName> ::= 'OFFSET='<Number>'&WIDTH='<Number>
<BlockConfig> ::= <BlockName>'&VALUE='<Number>
<RequestElement> ::= '&'<BlockName> | '&'<Label>
<NvConfig> ::= <Label>'='<String> | <Label>'='<Number>
```

Configuration strings

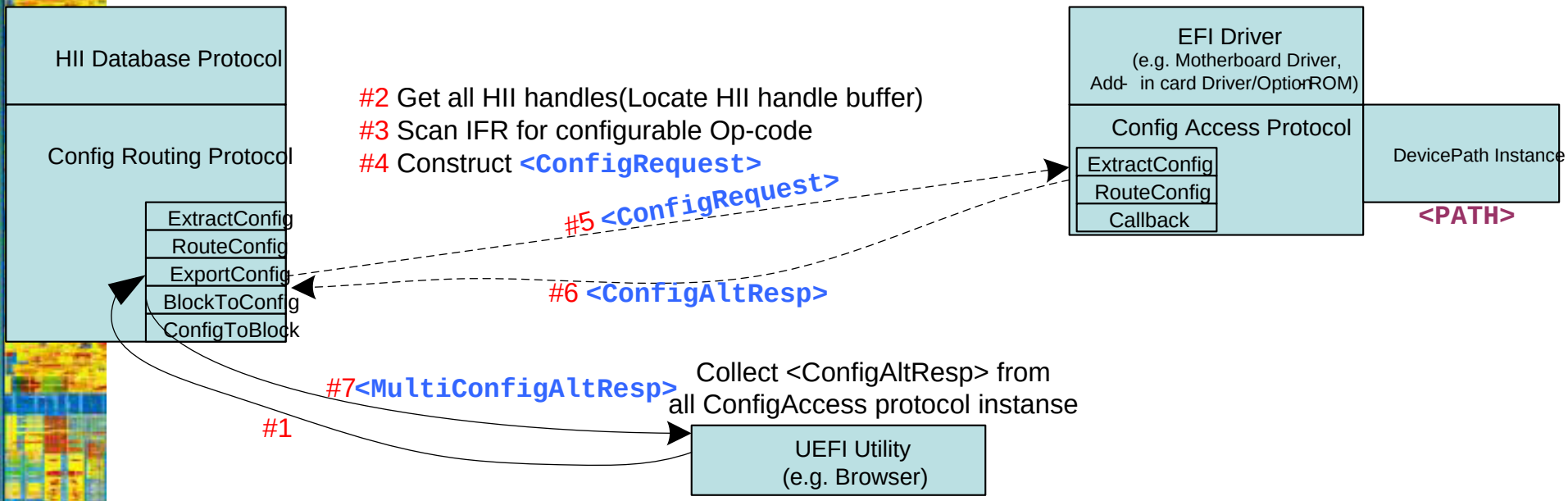
```
<ConfigRequest> ::= <ConfigHdr><RequestElement>*
<MultiConfigRequest> ::= <ConfigRequest>['&' <ConfigRequest>]*
<ConfigResp> ::= <ConfigHdr><ConfigBody>
<AltResp> ::= <AltConfigHdr><ConfigBody>
<ConfigAltResp> ::= <ConfigResp> ['&' <AltResp>]*
<MultiConfigAltResp> ::= <ConfigAltResp> ['&' <ConfigAltResp>]*
<MultiConfigResp> ::= <ConfigResp> ['&'<ConfigResp>]*
```

ConfigAccess/ConfigRouting Protocol



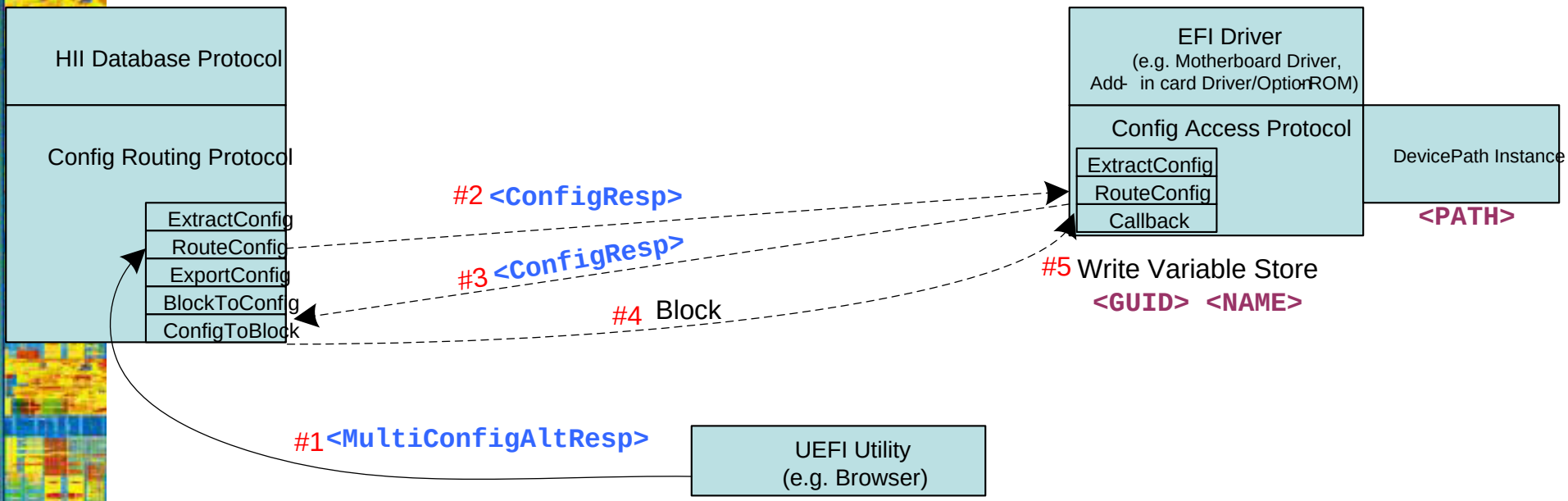
ConfigAccess/ConfigRouting Protocol

- ExportConfig



ConfigAccess/ConfigRouting Protocol

- RouteConfig



DISCLAIMER

- THIS INFORMATION COTNAINED IN THIS DOCUMENT, INCLUDING ANY TEST RESULTS ARE PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT OR BY THE SALE OF INTEL PRODUCTS. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. Intel products are not intended for use in medical, life saving, or life sustaining applications.

- Intel retains the right to make changes to its specifications at any time, without notice.
- Recipients of this information remain solely responsible for the design, sale and functionality of their products, including any liability arising from product infringement or product warranty.
- Intel may make changes to specifications, product roadmaps and product descriptions at any time, without notice.
- Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

- *Other names and brands may be claimed as the property of others.

-

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2®, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804