[UEFI Practice] HII Code Example

原创 jiangwei0512 Modified on 2022-02-14 23:37:31 Read 7.5k Collection 45 Likes 14

Category Column: UEFI Development Basics Article Tags: hii setup uefi

摘要 This article details how to build a graphical interface in a UEFI environment, including initializing the HII package, displaying a graphical page, and creating various interactive elements such as static text, selection boxes, string input, numeric input, and jumps. It also shows how to handle user input, save configurations, and perform backend interactions through EFI_HII_CONFIG_ACCESS_PROTOCOL. The content covers the entire process from VFR files to actual UI display.

The summary is generated in C Know , supported by DeepSeek-R1 full version, go to experience>

Code Sample

The code is at https://gitee.com/jiangwei0512/edk2-beni , and the module is BeniPkg\DynamicCommand\SetupDynamicCommand\SetupDynamicCommand.inf. Here, a command setup is used to open the graphical interface . The form of the graphical interface is in Page.vfr, and there are several uni files to store strings, which are initialized by the following code:

c AI generated projects 登录复制 run

```c
EFI_HII_HANDLE
InitializeHiiPackage (
  IN  EFI_HANDLE                    ImageHandle
  )
{
  EFI_STATUS                        Status;
  EFI_HII_PACKAGE_LIST_HEADER       *PackageList;
  EFI_HII_HANDLE                    HiiHandle;

  //
  // Retrieve HII package list from ImageHandle.
  //
  Status = gBS->OpenProtocol (
                ImageHandle,
                &gEfiHiiPackageListProtocolGuid,
                (VOID **)&PackageList,
                ImageHandle,
                NULL,
                EFI_OPEN_PROTOCOL_GET_PROTOCOL
                );
  if (EFI_ERROR (Status)) {
    return NULL;
  }

  //
  // Publish HII package list to HII Database.
  //
  Status = gHiiDatabase->NewPackageList (
                         gHiiDatabase,
                         PackageList,
                         NULL,
                         &HiiHandle
                         );
  if (EFI_ERROR (Status)) {
    return NULL;
  }

  return HiiHandle;
}
```

收起

Here we use the method of putting resources into the binary. Then we use the following code to display the graphics:

c AI generated projects 登录复制 run

```c
VOID
DisplayPage (
  VOID
  )
{
  EFI_STATUS                  Status;
  EFI_BROWSER_ACTION_REQUEST  ActionRequest;

  Status       = EFI UNSUPPORTED;
```

展开

formset

The content of the Page.vfr file used at the beginning:

json AI generated projects 登录复制

```json
// {76B732B8-B777-4ECF-A84E-7A8CA2484555}
#define FORMSET_GUID        { 0x76b732b8, 0xb777, 0x4ecf, 0xa8, 0x4e, 0x7a, 0x8c, 0xa2, 0x48, 0x45, 0x55 }

formset
  guid      = BENI_FORMSET_GUID,
  title     = STRING_TOKEN(STR_PAGE_TITLE),
  help      = STRING_TOKEN(STR_EMPTY_STRING),
  classguid = BENI_FORMSET_GUID,
```

```
10
endformset;
```

收起 ^

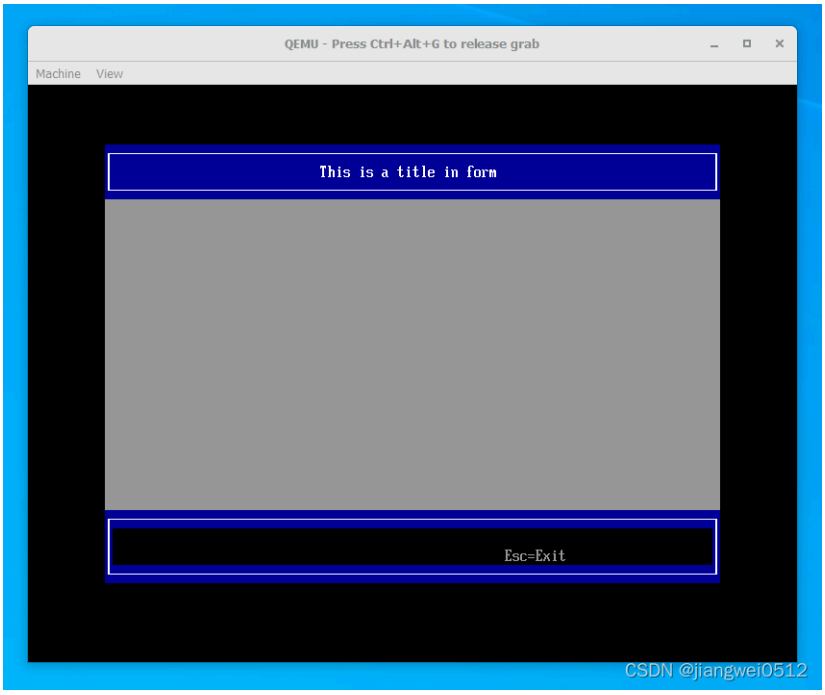
form

There is only one formset, and nothing will be displayed at this time. You need to add content to it. First, a form:

```json
// {76B732B8-B777-4ECF-A84E-7A8CA2484555}
#define BENI_FORMSET_GUID   { 0x76b732b8, 0xb777, 0x4ecf, 0xa8, 0x4e, 0x7a, 0x8c, 0xa2, 0x48, 0x45, 0x55 }
#define FRONT_PAGE_FORM_ID  0x1000

formset
  guid      = BENI_FORMSET_GUID,
  title     = STRING_TOKEN(STR_PAGE_TITLE_FORMSET),
  help      = STRING_TOKEN(STR_EMPTY_STRING),
  classguid = BENI_FORMSET_GUID,

  form
    formid  = FRONT_PAGE_FORM_ID,
    title   = STRING_TOKEN(STR_PAGE_TITLE_FORM);

  endform;

endformset;
```

收起 ^

The result obtained at this time is:



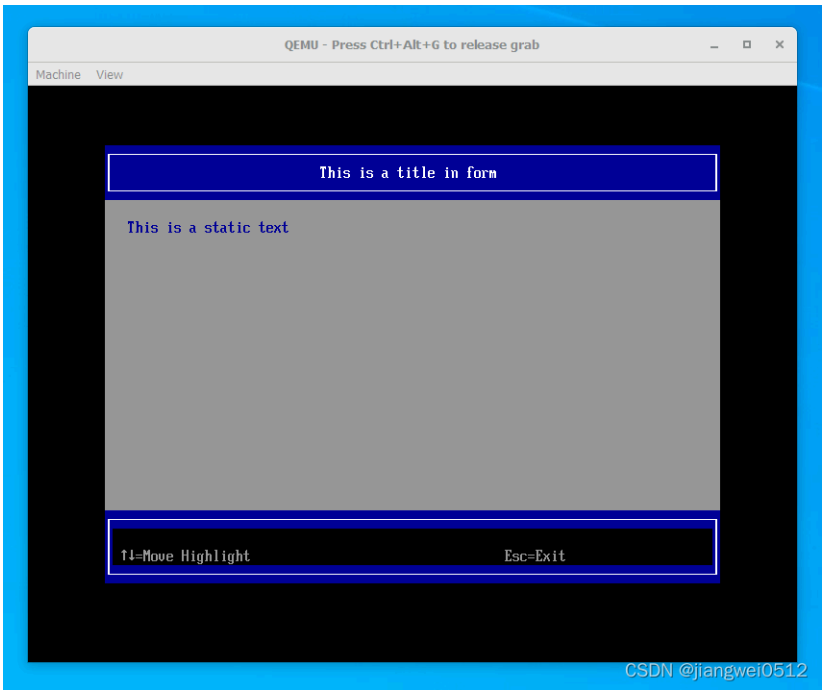
The title comes from `STR_PAGE_TITLE_FORM`, `Esc=Exit` but is `SendForm()` self-generated.

subtitle

Then you can add content to the form. First add a static string:

```json
  form
    formid  = FRONT_PAGE_FORM_ID,
    title   = STRING_TOKEN(STR_PAGE_TITLE_FORM);

    subtitle text = STRING_TOKEN(STR_PAGE_STATIC_TEXT);

  endform;
```

The result is:



You can see that `SendForm()` you have generated one `↑↓=Move Highlight` .

oneof

Then add a checkbox and its associated variables as follows:

json AI generated projects 登录复制

```
  efivarstore BENI_SETUP_DATA,
    attribute = 0x2, // EFI_VARIABLE_BOOTSERVICE_ACCESS
    name  = BeniSetupData,
    guid  = BENI_FORMSET_GUID;

  form
    formid  = FRONT_PAGE_FORM_ID,
    title   = STRING_TOKEN(STR_PAGE_TITLE_FORM);

    subtitle text = STRING_TOKEN(STR_PAGE_STATIC_TEXT);

    oneof varid = BeniSetupData.Data1,
      prompt      = STRING_TOKEN(STR_SELECT_DATA_1_PROMPT),
      help        = STRING_TOKEN(STR_SELECT_DATA_1_HELP),
      flags       = NUMERIC_SIZE_1 | INTERACTIVE | RESET_REQUIRED,
      option text = STRING_TOKEN(STR_SELECT_DATA_0), value = 0, flags = DEFAULT;
      option text = STRING_TOKEN(STR_SELECT_DATA_1), value = 1, flags = 0;
    endoneof;
    oneof varid = BeniSetupData.Data2,
      prompt      = STRING_TOKEN(STR_SELECT_DATA_2_PROMPT),
      help        = STRING_TOKEN(STR_SELECT_DATA_2_HELP),
      flags       = NUMERIC_SIZE_1 | INTERACTIVE | RESET_REQUIRED,
      option text = STRING_TOKEN(STR_SELECT_DATA_0), value = 0, flags = DEFAULT;
      option text = STRING_TOKEN(STR_SELECT_DATA_1), value = 1, flags = 0;
    endoneof;

  endform;
```

收起 ^

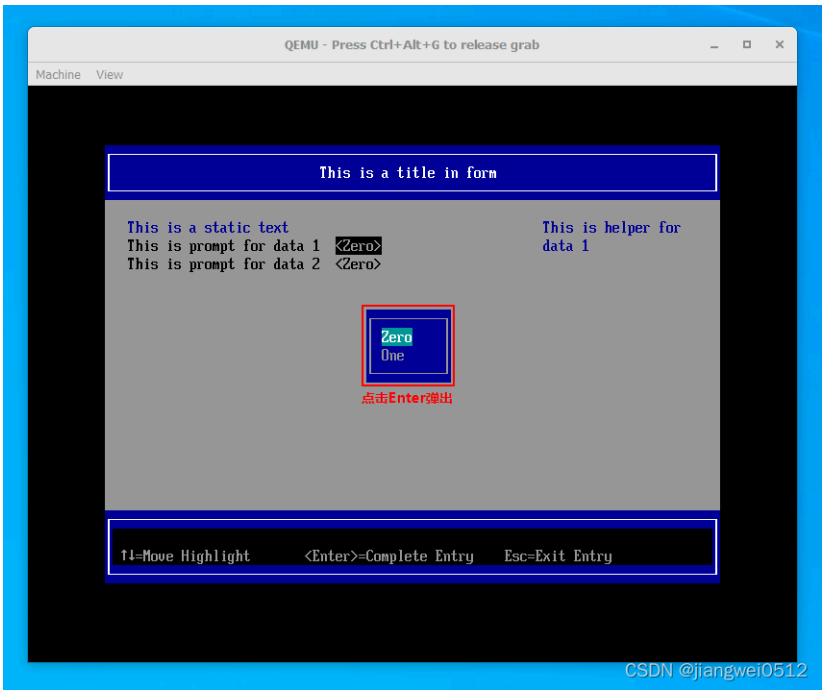
The corresponding variable structure:

c AI generated projects 登录复制 run

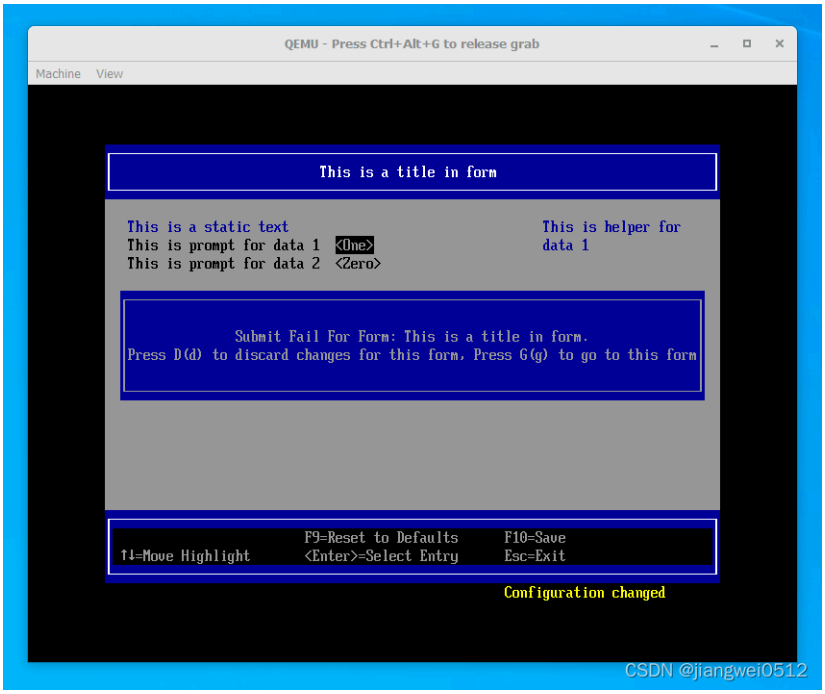
```
//
// This is used in name of efivarstore.
//
#define BENI_SETUP_DATA_VAR_NAME    L"BeniSetupData"

typedef struct {
  UINT8    Data1;
  UINT8    Data2;
  UINT8    Rsvd1[2];
} BENI_SETUP_DATA;
```

收起 ^

The results are as follows:



The value can be modified and saved here, but because there is no code implementation in the backend, an error will be reported:



Therefore, it is also necessary to increase the backend code, which mainly includes several parts: variable initialization, implementation and installation of EFI_HII_CONFIG_ACCESS_PROTOCOL.

Here we first initialize the corresponding variables in vfr:

c AI generated projects 登录复制 run

```c
EFI_STATUS
PrepareData (
  VOID
  )
{
  EFI_STATUS         Status;
  BENI_SETUP_DATA    *Data;
  UINTN              DataSize;

  Status    = EFI_UNSUPPORTED;
  Data      = NULL;
  DataSize  = sizeof (BENI_SETUP_DATA);

  Data = AllocateZeroPool (DataSize);
  if (NULL == Data) {
    DEBUG ((EFI_D_ERROR, "[BENI]%a %d Out of memory\n", __FUNCTION__, __LINE__));
    return EFI_OUT_OF_RESOURCES;
  }

  Status = gRT->GetVariable (
                  BENI_SETUP_DATA_VAR_NAME,
                  &gBeniSetupFormSetGuid,
                  NULL,
                  &DataSize,
                  Data
```

```
                );
  if (EFI_ERROR (Status)) {
    if (EFI_NOT_FOUND == Status) {
      DEBUG ((EFI_D_ERROR, "[BENI]Initialize Setup data\n"));
      Data->Data1 = 1;
      Data->Data2 = 1;
      DataSize    = sizeof (BENI_SETUP_DATA);
      Status = gRT->SetVariable (
                    BENI_SETUP_DATA_VAR_NAME,
                    &gBeniSetupFormSetGuid,
                    EFI_VARIABLE_BOOTSERVICE_ACCESS,
                    DataSize,
                    Data
                    );
      DEBUG ((EFI_D_ERROR, "[BENI]Status: - %r\n", Status));
    }
  }

  return Status;
}
```

收起 ^

This itself is not very meaningful, it just initializes and sets a variable, the value of the variable is 1 (so it is no longer displayed as Zero, but One), which will also be reflected in the interface. Then install EFI_HII_CONFIG_ACCESS_PROTOCOL:

c — AI generated projects — 登录复制 — run

```c
mPrivateData->ConfigAccess.ExtractConfig  = ExtractConfig;
mPrivateData->ConfigAccess.RouteConfig    = RouteConfig;
mPrivateData->ConfigAccess.Callback       = DriverCallback;

//
// Publish sample formset.
//
Status = gBS->InstallMultipleProtocolInterfaces (
                &mPrivateData->DriverHandle,
                &gEfiDevicePathProtocolGuid,
                &mHiiVendorDevicePath,
                &gEfiHiiConfigAccessProtocolGuid,
                &mPrivateData->ConfigAccess,
                NULL
                );
```

收起 ^

The functions here `DriverCallback()` can be implemented according to actual conditions. Currently, only printing information is added. When the above selection box is operated, it will be called and the information will be output.

string

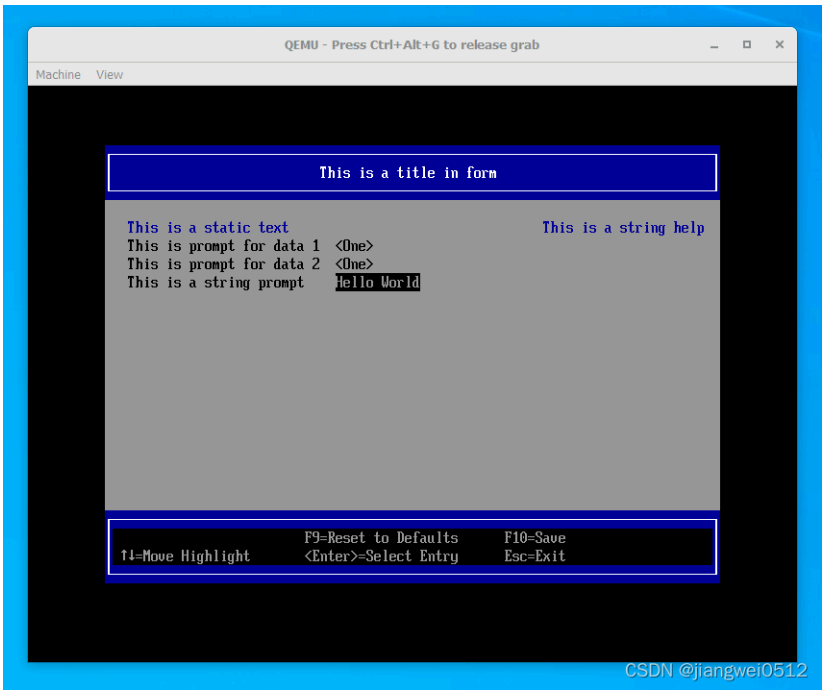
string is an editable string that can be saved to a variable after editing. The following is an example:

json — AI generated projects — 登录复制

```json
  string varid = BeniSetupData.DriverDescriptionData,
    questionid = PAGE_DESCRIPTION_ID,
    prompt     = STRING_TOKEN(STR_STRING_DESC_PROMPT),
    help       = STRING_TOKEN(STR_STRING_HELPER),
    flags      = INTERACTIVE,
    minsize    = 6,
    maxsize    = 30,
  endstring;
```

`DriverDescriptionData` is a member of a variable `BeniSetupData` , which can also be pre-initialized (in this case to "Hello World"), `PAGE_DESCRIPTION_ID` can be located `EFI_HII_CONFIG_ACCESS_PROTOCOL` in `Callback()` , and has some help information, size and operation limits, etc.

Here is the result:



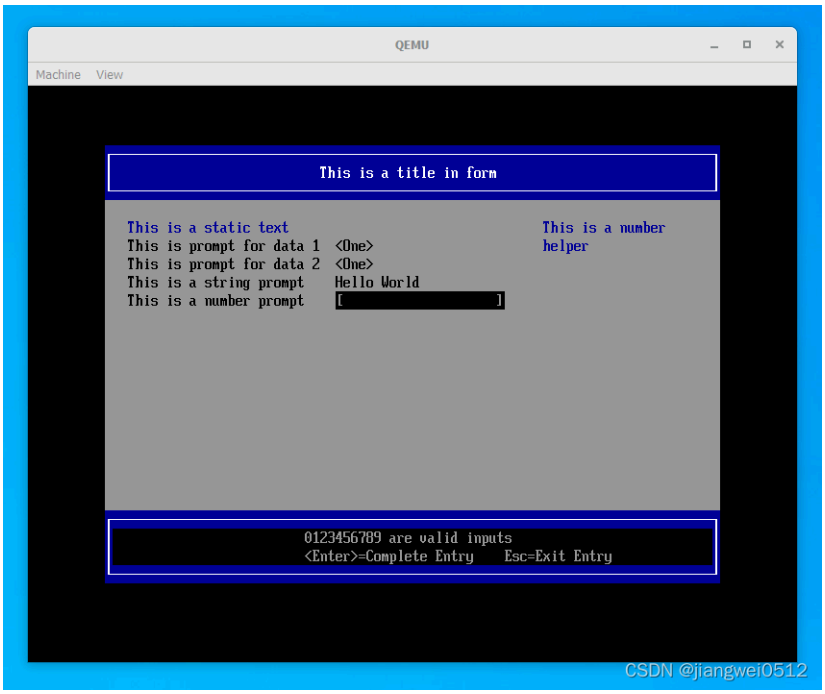
numeric

Nothing much to say, just numbers:

json AI generated projects 登录复制

```
    numeric varid   = BeniSetupData.Id,
            prompt  = STRING_TOKEN(STR_NUMERIC_ID_PROMPT),
            help    = STRING_TOKEN(STR_NUMERIC_ID_HELPER),
            minimum = 0,
            maximum = 1024,
    endnumeric;
```

Here are the results:



Similar to string, except that only numbers can be entered. Through `flag` configuration, you can choose to use decimal or hexadecimal.

text

Unlike subtitle, text can be selected. Here is an example:

json AI generated projects 登录复制

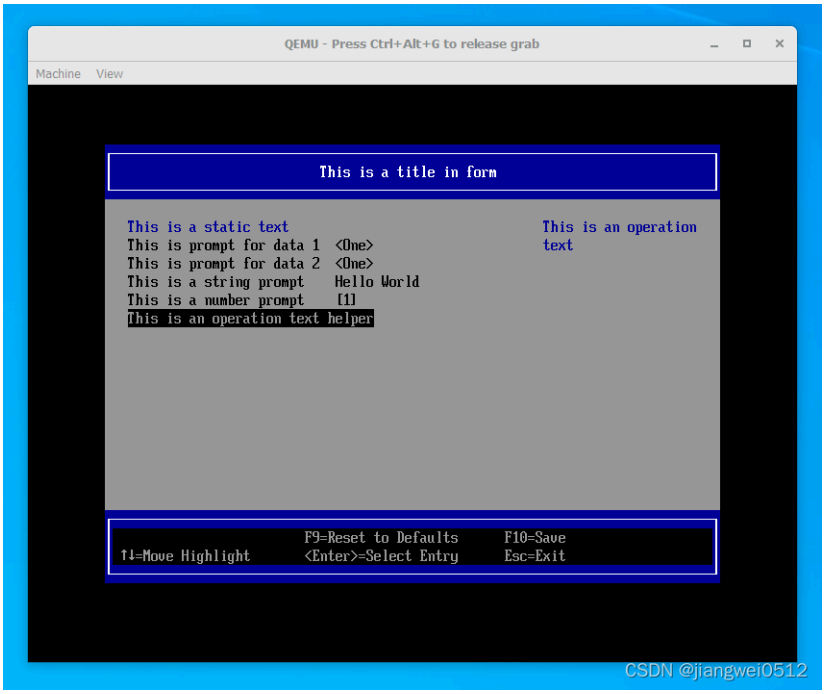
```
    text
      help   = STRING_TOKEN(STR_TEXT_PROMPT),
      text   = STRING_TOKEN(STR_TEXT_HELPER),
      flags  = INTERACTIVE,
      key    = PAGE_TEXT_ID;
```

`PAGE_TEXT_ID` Use in `EFI_HII_CONFIG_ACCESS_PROTOCOL : Callback()`

```c
EFI_STATUS
EFIAPI
DriverCallback (
  IN  CONST EFI_HII_CONFIG_ACCESS_PROTOCOL    *This,
  IN  EFI_BROWSER_ACTION                      Action,
  IN  EFI_QUESTION_ID                         QuestionId,
  IN  UINT8                                   Type,
  IN  EFI_IFR_TYPE_VALUE                      *Value,
  OUT EFI_BROWSER_ACTION_REQUEST              *ActionRequest
  )
{
  BENI_MODULE_START

  if (Action == EFI_BROWSER_ACTION_CHANGING) {
    switch (QuestionId) {
      case PAGE_TEXT_ID:
        DEBUG ((DEBUG_ERROR, "%a %d PAGE_TEXT_ID\n", __FUNCTION__, __LINE__));
        break;
      default:
        break;
    }
  } else if (Action == EFI_BROWSER_ACTION_CHANGED) {
    switch (QuestionId) {
      case PAGE_TEXT_ID:
        DEBUG ((DEBUG_ERROR, "%a %d PAGE_TEXT_ID\n", __FUNCTION__, __LINE__));
        break;
      default:
        break;
    }
  }

  BENI_MODULE_END
  return EFI_SUCCESS;
}
```

收起 ^

The display is as follows:



You can see `text` that the line can be selected, and after clicking it, you can see the print information:

```bash
[BENI]DriverCallback start...
DriverCallback 138 PAGE_TEXT_ID
[BENI]DriverCallback end...
[BENI]DriverCallback start...
DriverCallback 146 PAGE_TEXT_ID
[BENI]DriverCallback end...
```

The reason why this line can be operated is mainly because `flags = INTERACTIVE,` it will create an `EFI_IFR_ACTION` operation code, which is equivalent to implanting an operable action.

checkbox

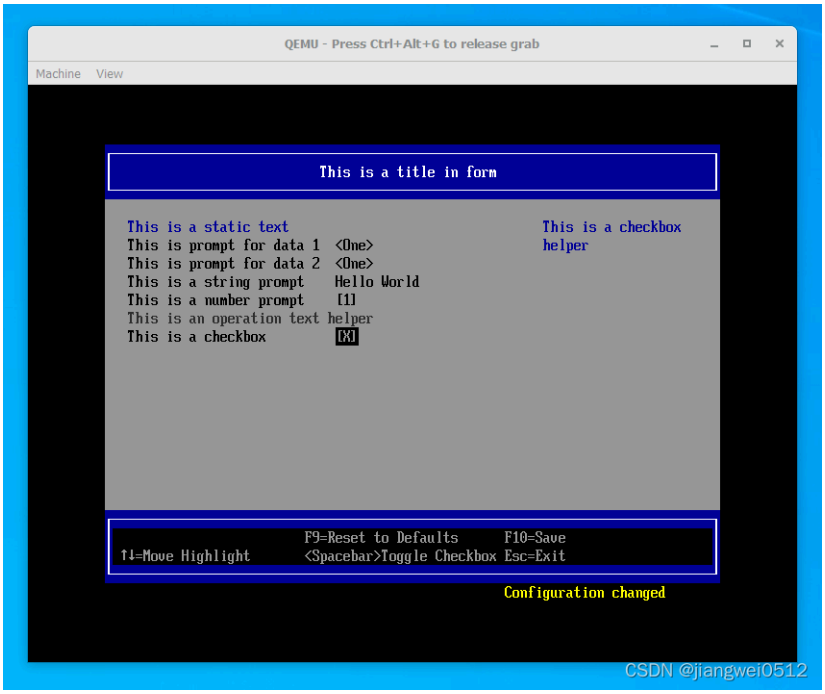
A checkbox can only have two values: TRUE and FALSE, or 0 and 1. Here is an example:

```bash
    grayoutif ideqval BeniSetupData.Disabled == 1;
      text
        help   = STRING_TOKEN(STR_TEXT_PROMPT),
        text   = STRING_TOKEN(STR_TEXT_HELPER),
        flags  = INTERACTIVE,
        key    = PAGE_TEXT_ID;
    endif;
```

```
8
9
10    checkbox varid   = BeniSetupData.Disabled,
11             prompt   = STRING_TOKEN(STR_CHECKBOXK_PROMPT),
12             help     = STRING_TOKEN(STR_CHECKBOXK_HELPER),
13             flags    = CHECKBOX_DEFAULT,
      endcheckbox;
```

收起 ^

It is also used here `grayoutif` . After selecting it, the one used in the previous test `text` will become gray, as shown below:



goto

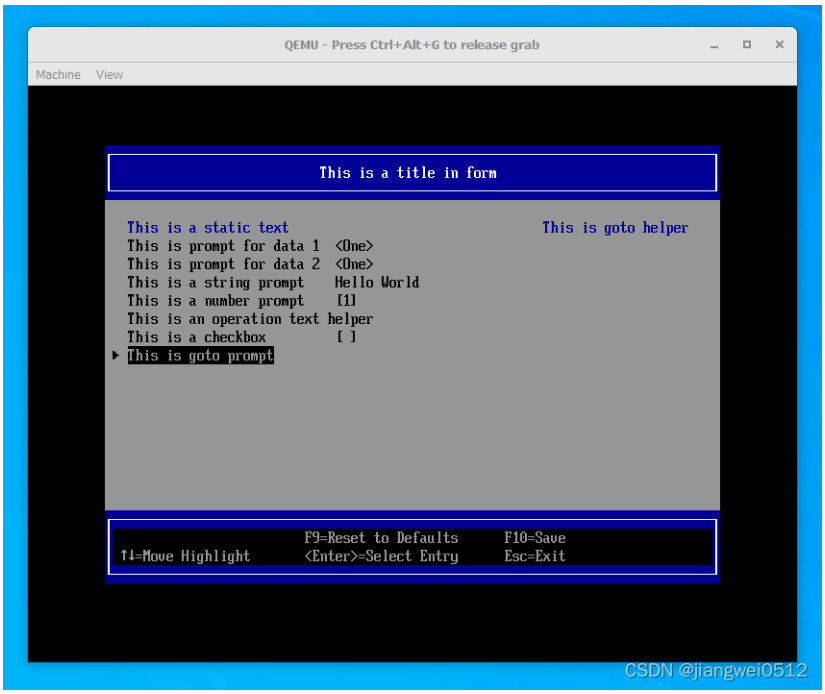
Used to jump to another interface:

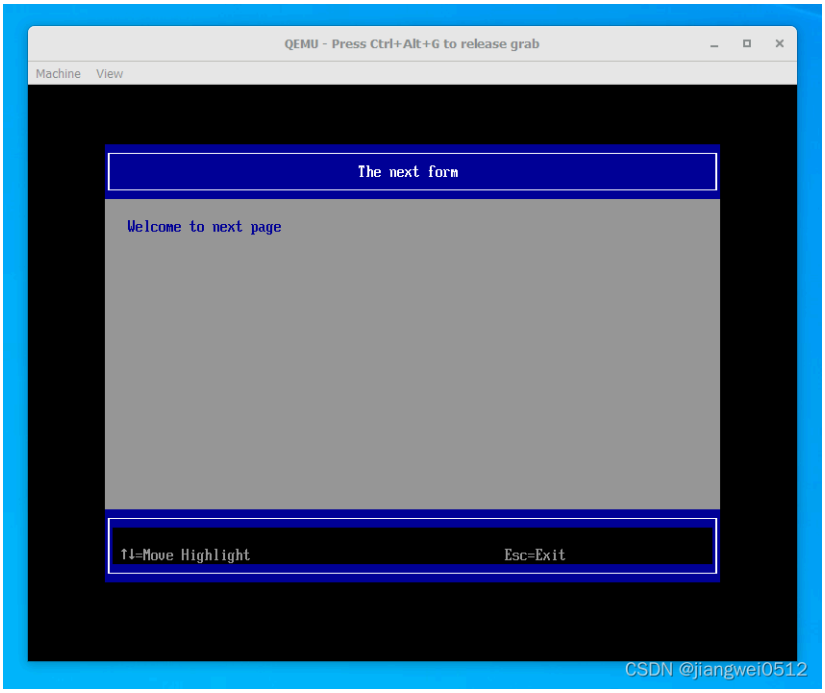
json AI generated projects 登录复制

```
1     goto PAGE_FORM_ID_2,
2       prompt  = STRING_TOKEN(STR_GOTO_PROMPT),
3       help    = STRING_TOKEN(STR_GOTO_HELPER);
4
5   endform;
6
7   form
8     formid  = PAGE_FORM_ID_2,
9     title   = STRING_TOKEN(STR_PAGE_TITLE_FORM_2);
10
11    subtitle text = STRING_TOKEN(STR_PAGE_NETX_PAGE);
12
13  endform;
```

收起 ^

Results displayed:





label

A label is equivalent to a placeholder in a VFR. It does not generate any displayable content by itself, but needs to be dynamically added through code. The specific method of adding content is to use `HiiCreateXXX()` the function introduced earlier to add form components. The following is an example of a label:

json AI generated projects 登录复制

```
#define LABEL_START                 0x1004
#define LABEL_END                   0x1005

  form
    formid  = PAGE_FORM_ID_2,
    title   = STRING_TOKEN(STR_PAGE_TITLE_FORM_2);

    subtitle text = STRING_TOKEN(STR_PAGE_NETX_PAGE);
    subtitle text = STRING_TOKEN(STR_EMPTY_STRING);

    label LABEL_START;
    label LABEL_END;

  endform;
```

收起 ^

You can see that only two are added here `label`, and the real operation is still in the code:

c AI generated projects 登录复制 run

```
/**
  Customize menus in the page.

  @param[in]  HiiHandle             The HII Handle of the form to update.
  @param[in]  StartOpCodeHandle     The context used to insert opcode.

```

```
  @retval  NA

**/
VOID
CustomizePage (
  IN  EFI_HII_HANDLE               HiiHandle,
  IN  VOID                         *StartOpCodeHandle
  )
{
  //
  // Add OpCode here.
  //
  HiiCreateSubTitleOpCode (StartOpCodeHandle, STRING_TOKEN (STR_TEXT_IN_CODE), 0, 0, 0);
}

/**
  Update components.

  @param  NA

  @retval  NA

**/
VOID
UpdatePageForm (
  VOID
  )
{
  VOID                        *StartOpCodeHandle;
  VOID                        *EndOpCodeHandle;
  EFI_IFR_GUID_LABEL          *StartGuidLabel;
  EFI_IFR_GUID_LABEL          *EndGuidLabel;

  //
  // Allocate space for creation of UpdateData Buffer
  //
  StartOpCodeHandle = HiiAllocateOpCodeHandle ();
  ASSERT (StartOpCodeHandle != NULL);

  EndOpCodeHandle = HiiAllocateOpCodeHandle ();
  ASSERT (EndOpCodeHandle != NULL);

  //
  // Create Hii Extend Label OpCode as the start opcode
  //
  StartGuidLabel = (EFI_IFR_GUID_LABEL *) HiiCreateGuidOpCode (StartOpCodeHandle, &gEfiIfrTianoGuid, NULL, sizeof (EFI_IFR_GUID_LABEL));
  StartGuidLabel->ExtendOpCode = EFI_IFR_EXTEND_OP_LABEL;
  StartGuidLabel->Number       = LABEL_START;
  //
  // Create Hii Extend Label OpCode as the end opcode
  //
  EndGuidLabel = (EFI_IFR_GUID_LABEL *) HiiCreateGuidOpCode (EndOpCodeHandle, &gEfiIfrTianoGuid, NULL, sizeof (EFI_IFR_GUID_LABEL));
  EndGuidLabel->ExtendOpCode = EFI_IFR_EXTEND_OP_LABEL;
  EndGuidLabel->Number       = LABEL_END;

  CustomizePage (
    mPrivateData->SetupHiiHandle,
    StartOpCodeHandle
    );

  HiiUpdateForm (
    mPrivateData->SetupHiiHandle,
    &gBeniSetupFormSetGuid,
    PAGE_FORM_ID_2,
    StartOpCodeHandle,
    EndOpCodeHandle
    );

  return;
}
```

收起 ^

The result is as follows, the red part is generated by code: