# EQP

- EQP__START.exe
  - > EQP_INF
    - EQP_WORKER.inf // 실행시킬 Object List를 정의한다.
    - EQP_METADATA.inf // Object 관련 정보를 설정한다.
    - EQP_SYSTEM.cfg // EQP_ID & RUN MODE를 설정한다.
    - EQP_IP.cfg // 각각의 EQP_ID 별로 IP & Port 정보를 설정한다.
    - > RES-ALARM
      - Alarm_db.alarm // 개별 Object 마다 Alarm Offset를 설정한다.
  - > EQP_PAGE
    - > CONFIG
      - CONFIG_PAGE.inf // EQP 내에서 사용할 Page 들을 등록한다.
    - > PROCESS
      - > MAIN // Process Recipe 폴더명은 User가 정의 할 수 있다.
        - *.rcp // Recipe File의 확장자는 rcp 입니다.
      - > PRE
      - > POST

EQP_WORKER.inf

```
// Command Format
// Command                Dll_Name          Class_Name          Object_Name
// Example
#OBJECT_STD               Dll_STD           CLS.xxx             OBJ_STD_01
 // -> ..₩EQP_BIN₩OBJ-STD₩ 폴더 내에서 해당 Dll를 찾는다.
#OBJECT_IO                Dll_IO            CLS.yyy             OBJ_IO_01
 // -> ..₩EQP_BIN₩OBJ-IO₩ 폴더 내에서 해당 Dll를 찾는다.
#OBJECT_INTERLOCK         Dll_INTERLOCK     CLS.zzz             OBJ_LOCK_01
 // -> ..₩EQP_BIN₩OBJ-INTERLOCK₩ 폴더 내에서 해당 Dll를 찾는다.
#OBJECT_LINK              EQP_LINK          C_01                OBJ_LINK_01
 // -> ..₩EQP_BIN₩OBJ-LINK₩ 폴더 내에서 해당 Dll를 찾는다.
#OBJECT_FA                Dll_FA            F_01                OBJ_FA_01
 // -> ..₩EQP_BIN₩OBJ-FA₩ 폴더 내에서 해당 Dll를 찾는다.
```

EQP_METADATA.inf

```
// Command Format
// Example
#OBJECT_PROPERTY      Object_Name

#DEFINE_CONSTANT      key_Word.01             Key_Data.01
#DEFINE_CONSTANT      key_Word.02             Key_Data.02
   ...
#FNC_MODE             Function_Name.01        Link_Object_Name    Object_Function_01
#FNC_MODE             Function_Name.02        Link_Object_Name    Object_Function_02
   ...
// 아래 Command는 Object Type이 STD인 경우에는 적용되지 않습니다.
#OBJECT_IO_FILE       IO_File_Name
  //  해당 파일은 ...₩EQP_INF₩[Object-Type]₩ 내에 있어야 합니다.
  // Object-Type이 IO 이면, OBJ-IO 이고
  // Object-Type이 LINK 이면, OBJ-LINK 이고
  // Object-Type이 FA이면, OBJ-FA가 해당 폴더가 됩니다.
#OBJECT_IO_PARA       IO.Key_Word.01          IO.Key_Data.01
#OBJECT_IO_PARA       IO.Key_Word.02          IO.Key_Data.02
   ...
```

EQP_SYSTEM.inf

```
// Command Format
// Example
#EQP_MODE    RUN       // Mode는 RUN 과 SIMULATION 이 있다.

#EQP_ID          0     // CTC   :  0
                       // TMC   :  1
                       // PM1   :  2
                       //    ...
                       // PM6   : 7
```

EQP_IP.inf

```
// Command Format
// Command    EQP_ID      IP              PORT
// Example
#EQP_INFO       0       127.0.0.1       10130
#EQP_INFO       1       127.0.0.1       10141
   ...
#EQP_INFO       6       127.0.0.1       10191
#EQP_INFO       7       127.0.0.1       10201
```

..₩RES-ALARM₩Alarm_db.alarm

```
// Command Format
// Command      Object_Name      Alarm_Offset
// Example
#OBJECT            OBJ_STD_01       1000
  // -> OBJ_STD_01 내에 정의된 모든 Alarm ID에 대해 Alarm 발생시 1000이 더해진 Alarm 값이 보고된다.
#OBJECT            OBJ_IO_01          2000
  // -> OBJ_IO_01 내에 정의된 모든 Alarm ID에 대해 Alarm 발생시 2000이 더해진 Alarm 값이 보고된다.

  ...
```

..₩CONFIG₩CONFIG_PAGE.inf

// Command Format
// config_page_name 만 입력하고, 확장자는 입력하지 않음.
// Example
Congile_Page_Name_01
Congile_Page_Name_02
...
Congile_Page_Name_XX

// UI Config Page과 Config.inf에 등록된 Page Name이 일치하지 않으면, Config Save Error Alarm이 발생됨.
// config 저장 파일의 확장자는 cfg 입니다.

..₩PROCESS₩

// Process Type 은 MAIN or PRE or POST의 3 종류가 있으며,
// 폴더 Name은 Process Type에 맞게 User가 변경 할 수 있습니다.
// 저장된 file의 확장자는 rcp 입니다.

**SUI**

-. SUI__START.exe // 실행 프로그램
-. SUI__EDITOR.exe // UI Page를 편집 할 수 있는 프로그램
-. MainFile.inf // 각각의 EQP 관련 UI 정보를 설정하는 정보 파일 입니다.

> **SCREEN_INF**

-. *.sct // user가 사용할 page 정보를 설정하는 파일입니다.
-. *.ctrl // page 제어 관련 정보를 설정하는 파일입니다.

> **SCREEN_RES**

> **PAGE**

-. *.page // user가 작성한 page 파일들이며, 확장자는 page 입니다.

> **RESOURCE**

> **GOI_BIN**

-. *.goi // user가 작성한 UI Part의 dll 확장자는 goi 입니다.

> **GOI_INF**

-. *.goi_inf // user가 작성한 UI Part의 정보 파일 확장자는 goi_inf 입니다.

> **IMG_RES**

-. *.bmp, *.jpg, *.png // page에서 사용할 image 자원을 모아둔다.

> **INCLUDE_FILE**

-. EQP_IP.cfg // EQP IP 정보 파일이며, EQP세서 사용하는 IP 정보 파일과 동일합니다.
// *.goi 내에서 사용되는 정보 파일은 이 폴더를 기준으로 합니다.

# ..₩*.page

```
#SCREEN_SIZE      page_width    page_height          // page size를 설정한다.

#BACK_PAGE_FILE                                    // background page command
{
   background_file                                  // background file 경로,  기준 폴더 : "..₩SCREEN_RES₩RESOURCE₩BACK_PAGE₩"
}

#GOBJ_INTERFACE                                    // gobj_interface command
{
   goi_info_file                                    // goi_info_ file 경로, 기준 폴더 : "..₩SCREEN_RES₩RESOURCE₩GOI_INF₩"
   sx  sy  ex  ey                                   // gobj part 위치 값
}
#RENAME                                            // 해당 part의 parameter
{
   keyword.01=data.01                               // userr가 정의한 keyword.01에 data.01이 설정된 상태입니다.
   keyword.02=data.02                               // userr가 정의한 keyword.02에 data.02이 설정된 상태입니다.
      …
}

#GP_IINF                                           // gp part command
{
   gp_file                                          //  gp file의 경로,  기준 폴더 : "..₩SCREEN_RES₩RESOURCE₩₩IMG_PART"
   sx  sy  ex  ey                                   // gp part 위치 값
}
#RENAME                                            // 해당 part의 parameter
{
   data.01                                          // 첫번째 parameter의 값이 data.01로 설정된 상태입니다.
   data.02                                          // 두번째 paramete의 값이 data.01로 설정된 상태입니다.
      …
}
```

..₩*.gbg // background file
// 이 파일 내용은 ui 시작시에 background graphic 자원에 한번 그려지고,
// 이 결과물은 Image 자원으로 저장된다.
// 이 후 background file을 포함한 page가 호출될 때 image 결과물만 그대로 다시 그린다.
// 화면 전환 속도를 빠르게 하기 위한 기능입니다.

```
#BACKGROUND
{
   img_file                // img_file  :  bmp,  jpg,  png 가 사용 가능함.
}
```

// page 작성시 사용된 command를 그대로 사용 가능함.

..₩*.goi // goi dll file
// user가 사용 목적에 맞게 만든 dll 파일입니다.

**// 아래의 Interface 함수를 통해 UI Engine에 의해 호출되고, 메모리가 할당됩니다.**

```
extern "C" __declspec(dllexport)
Obj__Interface_Ctrl* Create_GObj(const CString& str_gobj)
{
    ...
}
```

**// 아래의 Interface class를 통해 UI Engine와 user가 만든 goi가 연결됩니다.**

```
class GObj__Interface_Ctrl
{
public:
    virtual ~GObj__Interface_Ctrl(){};

    //-----------------------------------------------------------
    virtual void Get_Parameter(GObj__Parameter_Def* p_obj_para_def) = 0;

    virtual void Set_Parameter(const int& module_id,  const POINT& st,  const POINT& et,  const HWND& hwnd,  GObj__Resource_Info *p_obj_res) = 0;
    virtual void Hide_Draw(const HWND& hwnd) { };
    virtual void Show_GObj(const HWND& hwnd) = 0;
    virtual void Hide_GObj() = 0;

    //-----------------------------------------------------------
    virtual void WM_Paint(const HWND& hwnd) { };
    virtual void WM_Timer(const HWND& hwnd) { };

    virtual void WM_LButtonDown(const POINT& mouse_pt,const HWND& hwnd) { };
    virtual void WM_LButtonUp(const HWND& hwnd) { };

      ...

    virtual void WM_LButtonUp_With_Point(const POINT& mouse_pt,const HWND& hwnd) { };
   virtual void WM_RButtonUp_With_Point(const POINT& mouse_pt,const HWND& hwnd) { };
}
```

..₩*.goi_inf  // goi 관련 정보 파일입니다.
// 이 정보 파일은 goi 파일과 같이 배포되어야 합니다.

```
#GOI_INFO
{
   goi_file              // user가 만든 goi  dll 프로그램이며,  기준폴더는 "..₩SCREEN_RES₩RESOURCE₩GOI_BIN₩" 입니다.
   goi_class_name        // goi dll 내에 정의된 class name 입니다.
   sx  sy  ex  ey        // ui_editor page에 나타나는 goi part의 초기 위치 값 입니다.
}
```

..₩*.gp // 간단한 image 제어 script 프로그램입니다.

```
#BITMAP_IMAGE
   #FILE=img_file                    // image file의 경로이며,  기준 폴더는 "..₩SCREEN_RES₩RESOURCE₩IMG_RES" 입니다.
   #MASK_COLOR=RGB(0, 0, 0)          // 투명 처리할 Color를 설정합니다.
#CONTROL
   #CHANNEL    channel.01            // user 연결할 channel의 keyword를 만듭니다.
[조건문]                             // IF_AND:  or  IF_OR:  ...
VIS 1                                // 위의 조건문이 만족되면, 해당 image를  그립니다.
#END
```

# MainFile.inf (1)

**#MODULE_WINDOW_HIDE**

```
/*
  Default : Show
  이 기능이 활성화 되면, Window OS bottom bar에 EQP Module 수 만큼 window icon이 생성됩니다.
  권장 사항 : Hide
*/
```

**#SINGLE_LOCAL** EQP_ID

```
// 개별 EQP 테스트 시 Local IP (127.0.0.1)를 통해 UI와 일대일 연결된다.
// 동일한 PC에 EQP & UI 프로그램이 실행되어야 한다.
```

**#REMOTE_LOCAL** EQP_ID

```
// 개별 EQP 테스트 시 Remote IP를 통해 UI와 일대일 연결된다.
// EQP & UI 프로그램은 원격에 있는 PC에서 실행 될 수 있다.
```

**#SYSTEM_USER_LEVEL** User_Level

```
//  User_Level은 UNKNOWN  OPERATE  MAINT  PROCESS  MANAGER  OEM 등이 있습니다.
```

# MainFile.inf (2)

```
#ACTIVE.POPUP_PAGE_REDRAW
/*
  Default : Disable
  이 기능은 Popup Type의 Page가 나타날 때 Page를 다시 그리는 방식이며, Memory 사용을 낮출 수 있습니다.
  이 옵션이 활성화 되지 않으면, UI 프로그램 시작될 때 Memory에 Loading 되어 보다 빠른 화면 전환이 이루어 집니다.
  권장 사항 : 활성화.
*/

#EVENT_LEFT_BUTTON_UP_ENABLE
/*
  Default : Disable
  이 기능이 활성화 되면, Left Mouse Click시 Down에서 동작되지 않고 Up시 동작 됩니다.
  권장 사항 :활성화 권장, 양상 라인에서 명령 실행시 한번 더 생각 할 Time이 있어 사고 예방 차원.
*/

#WIN_EQPn          // n (EQP_ID) : 0 ~ n
/*
  #WIN_EQP0 와 #MAIN_DATA 는 동일한 명령어 입니다.
  #WIN_EQP1 와 #PM0_DATA 는 동일한 명령어 입니다.
  #WIN_EQP2 와 #PM1_DATA 는 동일한 명령어 입니다.
  ...
*/
{
   #SEQ_NAME       XXX     // EQP 목적에 부합되는 이름을 설정합니다. Ex) CTC, TMC, PMC ...

   #SCREEN_FILE        SCR_File_Name     // Page 관련 정보 파일이며, 경로는 ..₩SCREEN_INF 폴더가 기준이 됩니다.
   #SCREEN_CTRL    CTRL_File_Name       // Page 제어 정보 파일이며, 경로는 ..₩SCREEN_INF 폴더가 기준이 됩니다.

   #DIR_SEQ_ROOT    Eqp_Full_Path       // EQP 실행 프로그램이 있는 Full Path 경로와 일치해야 합니다.
}
```

*.sct

```
#SCREEN_DIR      Page_Sub_Folder
/*
  개별 EQP 에 맞는 Page 들을 분류하기 위한 Folder 입니다.
*/

#DEFINE      Def_Name                    // User가 정의한 Defile Name
   POPUP        Popup_Type                // CTC :  Window Y 값이 0 부터 시작되는 고정 Window.
                                          // NO  :  Window Y 값이 100 부터 시작되는 고정 Window.
                                          // YES :   Title Bar가 있는 Popup Window, 해당 EQP UI로 전환될 때만 Popup이 나타남.
                                          // TOP_MOST :  Title Bar가 있는 Popup Window.  EQP UI와 관련 없이 항상 Popup 됨.
   LOW_LEFT_X       x_pos                 // Window의  시작 x 좌표 ( 단위 : pikel )
   LOW_LEFT_Y       y_pos                 // Window의  시작 y 좌표 ( 단위 : pikel )
   WIDTH          win_size                // Window의  Width
   HEIGHT         win_height              // Window의  Height
#END                                      // Define 종료

WINDOW_NAME     page_name                 // UI에서 사용할 User가 만든 page의 이름
{
   #CALL_DEFINE      Def_Name             // 위에서 정의된 Def_Name

   CTRL_CHANNEL     ui_control_channel    // 해당 Window 의 Show & Hide를 결정하는 Channel
                                             // ui control channel의 값이 해당 window name과 일치하면 나타나고,
                                             //  일치하지 않으면, 사라집니다.
                                             // eqp channel 의 command는 SEQ_CTRL_CHANNEL 입니다.
   USER_LEVEL         page_level          // UMKNOWN   OPERATE   MAINT   PROCESS   MANAGER   OEM
}
```

*.ctrl

```
#GUI_CHANNEL__MODULE_INDEX      module_index              // EQP_ID 값을 나타내는UI Channel

#SEQ_CHANNEL__USER_LEVEL      eqp_user_level_channel      // user level 값을 나타내는 Channel을 연결한다.
#SEQ_CHANNEL__USER_ID         eqp_user_id_channel         // user id l값을 나타내는 Channel을 연결한다.


#CHANNEL          ui_channel                              // user가 사용할 ui channe를 등록한다.


#CHANGE_INIT                                              // UI 시작시 처음 한번만 호출되는 함수 입니다.

   #SET_DATA( ui_channel, data )                          // ui_channelI에 data 을 setting 한다.
   #SET_VARIABLE( ui_channel, ui_variable )               // ui_channelI에 ui_variable이 가진 값을 setting 한다.

#CHANGE_END

#CHANGE_START( ui_channel, data )                         // ui_channel 값이 data 일때, 해당 함수가 실행 됩니다.

   #SET_DATA( ui_channel, data )                          // ui_channelI에 data 을 setting 한다.
   #SET_VARIABLE( ui_channel, ui_variable )               // ui_channelI에 ui_variable이 가진 값을 setting 한다.

#CHANGE_END
```