netX 90 GPIO 제어를 위한 STM32 호스트 코드

생성일 : 2025-07-21 10:22

요약

1.

Task

-

netX 90 GPIO 제어를 위한 STM32 호스트 코드

netX 90의 GPIO를 STM32와 같은 호스트에서 제어하는 방법은 주로 **cifX API**와 **듀얼 포트 메모리(DPM)** 인터페이스를 통해 구현됩니다[1][2][3]. 다음은 구체적인 구현 방법과 코드 예제입니다.

1. 하드웨어 구성

netX 90과 STM32 간의 연결은 두 가지 방식으로 가능합니다[4][5]:

SPI 인터페이스 연결:

- SPM0_CLK (PA5)
- SPM0_MISO (PG9)
- SPM0_MOSI (PB5)
- SPM0_CS# (PD14)
- SPM0_DIRQ (PD15) - 데이터 인터럽트
- SPM0_SIRQ (PF3) - 동기 인터럽트

병렬 DPM 인터페이스:

- STM32 FMC를 사용한 16비트 병렬 연결[4]

2. 소프트웨어 구조

STM32에서 netX 90의 GPIO를 제어하기 위해서는 다음 소프트웨어 스택이 필요합니다[1][3]:

text

```
STM32 Application
↓
cifX API
↓
cifX/netX Toolkit
↓
```

```
Serial DPM Interface (SPI)
↓
netX 90 DPM
```

3. cifX API 초기화 코드

```c
#include "cifx_api.h"
#include "serial_dpm.h"

// 디바이스 인스턴스 구조체
DEVICEINSTANCE s_tDevInstance;

// SPI 통신 초기화
void OS_SpiInit(void* pvOSDependent)
{
    // STM32 SPI 초기화 코드
    // HAL_SPI_Init() 등 사용
}

// SPI 데이터 전송 함수
void OS_SpiTransfer(void* pvOSDependent, uint8_t* pbSend, uint8_t* pbRecv,
uint32_t ulLen)
{
    // STM32 HAL SPI 전송 코드
    HAL_SPI_TransmitReceive(&hspi1, pbSend, pbRecv, ulLen, HAL_MAX_DELAY);
}

// netX 90 초기화
int InitNetX90(void)
{
    int32_t lTkRet = CIFX_NO_ERROR;
    int iSerDPMType;

    // cifX 툴킷 초기화
    lTkRet = cifXTKitInit();
    if(CIFX_NO_ERROR != lTkRet)
        return -1;

    // Serial DPM 초기화
    if(SERDPM_UNKNOWN == (iSerDPMType = SerialDPM_Init(&s_tDevInstance)))
    {
        return -2; // Serial DPM 인식 실패
    }

    // 디바이스 추가
    lTkRet = cifXTKitAddDevice(&s_tDevInstance);
    if(CIFX_NO_ERROR != lTkRet)
    {
        cifXTKitDeinit();
        return -3;
    }
```

```
    return 0; // 성공
}
```

4. GPIO 제어 함수

netX 90의 GPIO는 다음과 같은 타입들이 있습니다[67]:

```
// GPIO 타입 정의
typedef enum {
    DRV_DIO_LINE_GPIO = 0x00000001ul,
    DRV_DIO_LINE_PIO = 0x00000002ul,
    DRV_DIO_LINE_MMIO = 0x00000003ul,
    DRV_DIO_LINE_HIF = 0x00000004ul
} DRV_DIO_LINE_E;

// GPIO 채널 제어 예제
int ControlNetX90_GPIO(uint32_t channel_id, int state)
{
    CIFXHANDLE hChannel;
    int32_t lRet;

    // 채널 열기
    lRet = xChannelOpen(NULL, "cifX0", 0, &hChannel);
    if(CIFX_NO_ERROR != lRet)
        return -1;

    // GPIO 상태 설정 (특정 mailbox 패킷 사용)
    if(state)
    {
        // GPIO 핀을 HIGH로 설정하는 패킷 전송
        // 구체적인 패킷 구조는 netX 90 펌웨어에 따라 다름
    }
    else
    {
        // GPIO 핀을 LOW로 설정하는 패킷 전송
    }

    // 채널 닫기
    xChannelClose(hChannel);

    return 0;
}
```

5. SPI 통신을 통한 DPM 직접 제어

더 저수준의 제어를 원한다면 SPI를 통해 직접 DPM에 접근할 수 있습니다[8][9]:

```c
// SPI DPM 명령어 구조
typedef struct {
    uint8_t command;    // MSB: 1=read, 0=write
    uint8_t addr_high;  // 주소 상위 바이트
    uint8_t addr_mid;   // 주소 중간 바이트
    uint8_t addr_low;   // 주소 하위 바이트
} SPI_DPM_COMMAND;

// DPM 메모리 쓰기
int WriteDPM(uint32_t address, uint8_t* data, uint32_t length)
{
    SPI_DPM_COMMAND cmd;

    cmd.command = 0x00 | ((address >> 16) & 0x7F); // Write command
    cmd.addr_high = (address >> 16) & 0xFF;
    cmd.addr_mid = (address >> 8) & 0xFF;
    cmd.addr_low = address & 0xFF;

    // SPI 전송
    HAL_GPIO_WritePin(SPI_CS_GPIO_Port, SPI_CS_Pin, GPIO_PIN_RESET);
    HAL_SPI_Transmit(&hspi1, (uint8_t*)&cmd, 4, HAL_MAX_DELAY);
    HAL_SPI_Transmit(&hspi1, data, length, HAL_MAX_DELAY);
    HAL_GPIO_WritePin(SPI_CS_GPIO_Port, SPI_CS_Pin, GPIO_PIN_SET);

    return 0;
}

// DPM 메모리 읽기
int ReadDPM(uint32_t address, uint8_t* data, uint32_t length)
{
    SPI_DPM_COMMAND cmd;

    cmd.command = 0x80 | ((address >> 16) & 0x7F); // Read command
    cmd.addr_high = (address >> 16) & 0xFF;
    cmd.addr_mid = (address >> 8) & 0xFF;
    cmd.addr_low = address & 0xFF;

    // SPI 전송
    HAL_GPIO_WritePin(SPI_CS_GPIO_Port, SPI_CS_Pin, GPIO_PIN_RESET);
    HAL_SPI_Transmit(&hspi1, (uint8_t*)&cmd, 4, HAL_MAX_DELAY);
    HAL_SPI_Receive(&hspi1, data, length, HAL_MAX_DELAY);
    HAL_GPIO_WritePin(SPI_CS_GPIO_Port, SPI_CS_Pin, GPIO_PIN_SET);

    return 0;
}
```

```c
#include "cifXUser.h"
#include <string.h>

#define DIO_SET_LINE_CONFIG     0x00B1  // 핀 모드 설정 (입력/출력)
```

```c
#define DIO_SET_LINE_STATE      0x00B2  // 핀 상태 설정 (HIGH/LOW)
#define MAILBOX_DEST            0x20    // DIO 기능 대상 주소
#define TIMEOUT_MS              1000

typedef enum {
    DRV_DIO_LINE_GPIO = 0x00000001ul,
    DRV_DIO_LINE_PIO  = 0x00000002ul,
    DRV_DIO_LINE_MMIO = 0x00000003ul,
    DRV_DIO_LINE_HIF  = 0x00000004ul
} DRV_DIO_LINE_E;

/**
 * @brief 지정된 netX90 DIO 라인을 HIGH 또는 LOW로 설정
 * @param channel_id : DIO 라인 번호 (예: MMIO7 -> 7)
 * @param state      : 1 = HIGH, 0 = LOW
 */
int ControlNetX90_GPIO(uint32_t channel_id, int state)
{
    CIFXHANDLE hChannel;
    int32_t lRet;

    // 1. 채널 열기 (channel 0 = 실시간 채널)
    lRet = xChannelOpen(NULL, "cifX0", 0, &hChannel);
    if (lRet != CIFX_NO_ERROR)
        return -1;

    // 2. 해당 DIO 라인을 OUTPUT으로 설정 (설정은 1회만 해도 무방)
    typedef struct {
        CIFX_PACKET_HEADER tHeader;
        uint32_t ulLine;       // 라인 번호
        uint32_t ulType;       // 라인 타입 (MMIO, GPIO 등)
        uint32_t ulFlags;      // 0 = OUTPUT, 1 = INPUT
    } __attribute__((packed)) DIO_SET_LINE_CONFIG_REQ;

    DIO_SET_LINE_CONFIG_REQ tCfg;
    memset(&tCfg, 0, sizeof(tCfg));

    tCfg.tHeader.ulDest  = MAILBOX_DEST;
    tCfg.tHeader.ulCmd   = DIO_SET_LINE_CONFIG;
    tCfg.tHeader.ulLen   = sizeof(tCfg);
    tCfg.ulLine          = channel_id;
    tCfg.ulType          = DRV_DIO_LINE_MMIO;
    tCfg.ulFlags         = 0;  // 출력 설정

    // 출력 설정 명령 전송
    lRet = xChannelPutPacket(hChannel, (CIFX_PACKET*)&tCfg, TIMEOUT_MS);
    if (lRet != CIFX_NO_ERROR) {
        xChannelClose(hChannel);
        return -2;
    }

    // 3. 상태(HIGH/LOW) 설정 패킷 구성
    typedef struct {
```

```
        CIFX_PACKET_HEADER tHeader;
        uint32_t ulLine;       // DIO 라인 번호
        uint32_t ulType;       // MMIO 타입
        uint32_t ulState;      // 0 = LOW, 1 = HIGH
    } __attribute__((packed)) DIO_SET_LINE_STATE_REQ;

    DIO_SET_LINE_STATE_REQ tReq;
    memset(&tReq, 0, sizeof(tReq));

    tReq.tHeader.ulDest  = MAILBOX_DEST;
    tReq.tHeader.ulCmd   = DIO_SET_LINE_STATE;
    tReq.tHeader.ulLen   = sizeof(tReq);
    tReq.ulLine          = channel_id;
    tReq.ulType          = DRV_DIO_LINE_MMIO;
    tReq.ulState         = (state != 0) ? 1 : 0;  // HIGH or LOW

    // 핀 상태 설정 명령 전송
    lRet = xChannelPutPacket(hChannel, (CIFX_PACKET*)&tReq, TIMEOUT_MS);
    if (lRet != CIFX_NO_ERROR) {
        xChannelClose(hChannel);
        return -3;
    }

    // 4. 채널 닫기
    xChannelClose(hChannel);
    return 0;
}
```

📥 호출 예제

```
// MMIO7 핀을 HIGH로 설정
ControlNetX90_GPIO(7, 1);
// MMIO7 핀을 LOW로 설정
ControlNetX90_GPIO(7, 0);
```

✅ 중요 사항 요약

항목	설명
`xChannelOpen()`	netX 90 의 채널을 연다 (`"cifX0"`, `0`)
`DIO_SET_LINE_CONFIG`	MMIO 핀을 **출력 모드**로 설정
`DIO_SET_LINE_STATE`	핀 상태를 HIGH/LOW로 설정
`channel_id`	설정하고자 하는 MMIO의 번호 (예: 7 = MMIO7)
`DRV_DIO_LINE_MMIO`	핀 타입 식별자 (MMIO의 경우 `0x03`)

기존에 제공된 `App_DemoApplication.c` 파일 구조와 초기화 흐름을 기반으로, 작성한 `ControlNetX90_GPIO()` 함수 (MMIO7 제어용)를 **기존 애플리케이션 흐름에 통합**하는 코드를 아래와 같이 제안드립니다.

🔧 1. 함수 추가: `ControlNetX90_GPIO()`

먼저, `App_DemoApplication.c` 파일 하단 또는 적절한 유틸리티 정의 섹션에 이 함수를 **그대로 추가**합니다. 기존 스타일에 맞춰 주석 처리와 정렬도 반영했습니다.

```
/*****************************************************************************
*******/
/*! Set a MMIO GPIO line to HIGH or LOW
* \param channel_id MMIO line index (e.g. 7 for MMIO7)
* \param state 0 = LOW, 1 = HIGH
* \return 0 on success, error code otherwise
*/
/*****************************************************************************
*******/
int ControlNetX90_GPIO(uint32_t channel_id, int state)
{
  CIFXHANDLE hChannel;
  int32_t lRet;

  // Open default Channel 0
  lRet = xChannelOpen(NULL, "cifX0", 0, &hChannel);
  if(lRet != CIFX_NO_ERROR)
    return -1;

  // Set direction to output (only needed once, cached by firmware)
  typedef struct {
    CIFX_PACKET_HEADER tHeader;
    uint32_t ulLine;
    uint32_t ulType;
    uint32_t ulFlags;
  } DIO_SET_LINE_CONFIG_REQ;

  DIO_SET_LINE_CONFIG_REQ tCfg;
  memset(&tCfg, 0, sizeof(tCfg));
  tCfg.tHeader.ulDest = 0x20;
  tCfg.tHeader.ulCmd  = 0xB1;  // DIO_SET_LINE_CONFIG
  tCfg.tHeader.ulLen  = sizeof(tCfg);
  tCfg.ulLine         = channel_id;
  tCfg.ulType         = DRV_DIO_LINE_MMIO;
  tCfg.ulFlags        = 0; // Output

  lRet = xChannelPutPacket(hChannel, (CIFX_PACKET*)&tCfg, 1000);
  if(lRet != CIFX_NO_ERROR)
  {
    xChannelClose(hChannel);
    return -2;
  }
```

```
  // Set pin state
  typedef struct {
    CIFX_PACKET_HEADER tHeader;
    uint32_t ulLine;
    uint32_t ulType;
    uint32_t ulState;
  } DIO_SET_LINE_STATE_REQ;

  DIO_SET_LINE_STATE_REQ tReq;
  memset(&tReq, 0, sizeof(tReq));
  tReq.tHeader.ulDest = 0x20;
  tReq.tHeader.ulCmd  = 0xB2;  // DIO_SET_LINE_STATE
  tReq.tHeader.ulLen  = sizeof(tReq);
  tReq.ulLine         = channel_id;
  tReq.ulType         = DRV_DIO_LINE_MMIO;
  tReq.ulState        = (state) ? 1 : 0;

  lRet = xChannelPutPacket(hChannel, (CIFX_PACKET*)&tReq, 1000);
  xChannelClose(hChannel);

  return (lRet == CIFX_NO_ERROR) ? 0 : -3;
}
```

📍 2. 호출 위치 추가

`App_CifXApplicationDemo()`의 주 루프 또는 초기 설정 직후에 제어를 넣을 수 있습니다. 예를 들어, `/** now the bus is running */` 주석 아래 루프 전에 MMIO7을 **초기 상태로 설정**하도록 아래와 같이 삽입할 수 있습니다:

```
/** now the bus is running */
// Set MMIO7 initial state to HIGH
if (0 != ControlNetX90_GPIO(7, 1)) {
  PRINTF("Failed to set MMIO7 HIGH!" NEWLINE);
} else {
  PRINTF("Successfully set MMIO7 HIGH at startup." NEWLINE);
}
```

또는 지속적으로 토글하고자 하는 경우, 주 루프 내 아래와 같이 주기적으로 상태를 변경하도록 구현도 가능합니다:

```
bool pinHigh = true;

while(g_tAppData.fRunning && lRet == CIFX_NO_ERROR)
{
  App_IODataHandler(&g_tAppData);
  lRet = App_AllChannels_PacketHandler(&g_tAppData);

  // 주기적인 MMIO7 토글 (예제용)
```

```
    ControlNetX90_GPIO(7, pinHigh ? 1 : 0);
    pinHigh = !pinHigh;

    OS_Sleep(1000); // 1초 간격
}
```

📦 전체 통합 흐름 요약

단계	설명
1	`ControlNetX90_GPIO()` 함수 정의 코드 추가
2	`App_CifXApplicationDemo()` 내 `while()` 루프 전 또는 루프 내부에 상태 제어 코드 삽입
3	컴파일 및 실행 시 netX90의 MMIO7 핀을 제어 가능

📎 기타 참고사항

- 이 코드는 netX90이 DIO 펌웨어를 포함하고 있으며, GPIO 핀에 출력 기능이 라이센스 및 설정상 허용된 상태임을 전제합니다.
- `ulDest = 0x20`은 netX의 DIO mailbox 대상 주소이며, 적절한 구성일 경우 사용 가능합니다 (Hilscher 펌웨어에서 사용됨).

필요하시면 **입력 모드 처리**, **인터럽트 기반 입력 감지**, 또는 다른 핀에 대한 동일 처리도 추가해 드릴 수 있습니다.

1. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/31784576/6989b80c-c01f-4b6c-89be-7dcbfffcfc84/App_DemoApplication.c

📚 참고 문서

- **Hilscher netX Toolkit**: `cifXUser.h`
- Hilscher Application Note: **"DIO over DPM"**
- **Hilscher Toolkit Documentation**: Hilscher Knowledge Base

필요하시면 GPIO, PIO, HIF 타입에 대한 설정 예제도 제공해드릴 수 있습니다.

6. STM32CubeMX 설정

netX 90과 통신하기 위한 STM32 설정105:

1. **SPI 설정:**
   - Mode: Master
   - Clock: up to 50MHz
   - Data Size: 8-bit
   - CPOL: Low
   - CPHA: 1Edge
2. **GPIO 설정:**

- CS 핀: Push-pull output
- IRQ 핀들: Input with pull-up

7. 사용 가능한 리소스

하드웨어:

- netSHIELD 90-RE 평가 보드 (STM32 Nucleo 호환)[1][12]

소프트웨어:

- I-CUBE-NETX STM32Cube 확장 패키지[1][10]
- cifX API 및 툴킷[3][9]
- Hilscher 공식 문서 및 예제[13][14]

이러한 방법들을 통해 STM32에서 netX 90의 GPIO를 효과적으로 제어할 수 있으며, 특히 산업용 실시간 이더넷 통신과 함께 GPIO 제어가 필요한 애플리케이션에 적합합니다.

1. https://www.st.com/en/partner-products-and-services/i-cube-netx.html
2. https://hilscher.atlassian.net/wiki/spaces/CIFXDRV/pages/80905280/CIFX+API
3. https://www.hilscher.com/fileadmin/cms_upload/en-US/Resources/pdf/cifX_netX-Device_Driver_Datasheet_07-2024_EN.pdf
4. https://hilscher.atlassian.net/wiki/spaces/NETX/pages/144194962/STM32+FMC+-+netx+90+DPM+interface
5. https://deepbluembedded.com/stm32-spi-tutorial/
6. https://www.hilscher.com/external/netXDriverDocumentation/group___d_i_o.html
7. https://www.hilscher.com/fileadmin/cms_upload/en-US/Resources/pdf/Single-Pair_Ethernet_with_netX_90_AN_03_EN.pdf
8. https://www.youtube.com/watch?v=IADI1vZXPn4
9. https://www.hilscher.com/fileadmin/user*upload/global/resources/special_ProductAnnouncements/Driver_and_driver_toolkit/cifX_netX_Toolkit-*_DPM_TK_11_EN.pdf
10. https://www.st.com/en/partner-products-and-services/i-nucleo-netx.html
11. https://hilscher.atlassian.net/wiki/spaces/NETX/pages/807075841/netSHIELD+NSHIELD+90-RE
12. https://www.hilscher.com/company/news/netshield-90-re-new-evaluation-board-for-stm32-nucleo-64-and-nucleo-144
13. https://www.hilscher.com/external/netXDriverDocumentation/
14. https://github.com/HilscherAutomation/netPI-netx-programming-examples/blob/master/examples/sources/PNS_simpleConfig.c
15. https://nexp.tistory.com/27
16. https://www.hilscher.com/fileadmin/cms_upload/de/Resources/pdf/Manual_netX_Program_Reference_Guide_Rev04.pdf
17. https://prosigi.tistory.com/73
18. https://www.takagi-c.com/dcms_media/other/netRAPID90_Datasheet_02-2019_GB.pdf
19. https://www.st.com/resource/en/user_manual/um3003-getting-started-with-the-stswsilplc-firmware-for-the-stevalsilplc01-solution-stmicroelectronics.pdf

20. https://www.hannovermesse.de/apollo/hannover_messe_2021/obs/Binary/A1090491/netX-90_Datasheet_10-2019_GB.pdf
21. https://www.hilscher.com/kr/%ED%9A%8C%EC%82%AC/%EB%89%B4%EC%8A%A4/netshield-90-re-stm32-nucleo-64%EC%99%80-nucleo-144%EB%A5%BC-%EC%9C%84%ED%95%9C-%EC%83%88%EB%A1%9C%EC%9A%B4-%ED%8F%89%EA%B0%80-%EB%B3%B4%EB%93%9C
22. https://www.st.com/resource/en/user_manual/um2808-getting-started-with-the-stswethdrv01v1-firmware-for-servo-drive-ethercat-solution-stmicroelectronics.pdf
23. https://hilscher.atlassian.net/wiki/spaces/NETX/pages/144200008/Using+3+UART+interfaces
24. https://www.youtube.com/watch?v=dkTK5-Vv1Fc
25. https://www.hilscher.com/kr/company/news/netshield-90-re-new-evaluation-board-for-stm32-nucleo-64-and-nucleo-144
26. https://www.hilscher.com/products/multiprotocol-socs-stacks/multiprotocol-socs/netx-90
27. https://www.embedded-office.com/services/netx-integration
28. https://vuzwa.tistory.com/entry/STM32-5-GPIO-%EC%A0%9C%EC%96%B4%ED%95%98%EA%B8%B0B-L475E-IOT01A1-%EA%B0%9C%EB%B0%9C%EB%B3%B4%EB%93%9C-%ED%99%9C%EC%9A%A9%ED%95%98%EA%B8%B0
29. https://hilscher.atlassian.net/wiki/spaces/NETX/pages/144199983/netX+90+peripheral+drivers+driver+examples+and+protocol+examples+links
30. https://hilscher.atlassian.net/wiki/spaces/NETX/pages/144183044
31. https://www.hilscher.com/fileadmin/cms_upload/en-US/Resources/pdf/netX-Broschuere-EN-Web-RZ_PDF.pdf
32. http://fs.gongkong.com/files/technicalData/200806/20080619131246000001.pdf
33. https://brtchip.com/wp-content/uploads/sites/3/2021/07/AN_372-FT90x-UART-to-GPIO-Bridge.pdf
34. https://www.youtube.com/watch?v=0wfAcb6_IfY
35. https://www.ftdichip.com/Support/Documents/AppNotes/AN_415%20FT90x%20Ethernet%20to%20GPIO%20Bridge.pdf
36. https://github.com/HilscherAutomation/netPI-netx-programming-examples
37. https://deepbluembedded.com/how-to-receive-spi-with-stm32-dma-interrupt/
38. https://www.hilscher.com/kr/%EC%A0%9C%ED%92%88/%EB%A9%80%ED%8B%B0-%ED%94%84%EB%A1%9C%ED%86%A0%EC%BD%9C-soc-%EB%B0%8F-%EC%8A%A4%ED%83%9D/%EB%A9%80%ED%8B%B0-%ED%94%84%EB%A1%9C%ED%86%A0%EC%BD%9C-soc/netx-90
39. https://www.youtube.com/watch?v=MgRqwNM5acY
40. https://www.ethercat.org/es/products/833ED4AC6D8C461EA3BAB65025B02C9C.htm
41. https://www.hilscher.com/fileadmin/user_upload/global/resources/special_ProductAnnouncements/Dual-port-memory/netX_Dual-Port_Memory_Interface_DPM_17_EN.pdf
42. https://hilscher.atlassian.net/wiki/spaces/NETX/pages/144196372/netX+90+in+Motion
43. https://uvadoc.uva.es/bitstream/handle/10324/32511/TFM-I-984%20anexos%20netX%20Dual-Port%20Memory%20Interface%20DPM%2012%20EN.pdf?sequence=5
44. https://hilscher.atlassian.net/wiki/spaces/GLOBALSUP/pages/122812845/CIFX+API+meets+C+.NET+EN
45. https://easyfairsassets.com/sites/266/2022/12/netX-Broschuere-EN-RZ.pdf

46. https://www.hilscher.com/fileadmin/cms_upload/de/Resources/pdf/PC_Cards_CIFX_50_50E_70E_100EH_UM_53_EN.pdf
47. https://www.youtube.com/watch?v=pqJz9JsVWWk
48. https://uvadoc.uva.es/bitstream/handle/10324/32511/TFM-I-984%20anexo%20cifX%20API%20PR%2003%20EN.pdf?sequence=2
49. https://hilscher.atlassian.net/wiki/x/HCSiB
50. https://www.profibus.fr/wp-content/uploads/2018/02/netX_90_DS_Datasheet_2017-01_GB.pdf
51. https://www.semitron.de/wp-content/uploads/2025/05/netSHIELD_NSHIELD_90-RE_UM_02_EN.pdf
52. https://en.wikipedia.org/wiki/Hilscher_netx_network_controller
53. https://www.hannovermesse.de/apollo/hannover_messe_2025/obs/Binary/A1432668/1432668_05069398.pdf
54. https://hilscher.atlassian.net/wiki/spaces/NETX/pages/144199392/netX+90+APP+side+Flash+Memory+programming
55. https://hilscher.atlassian.net/wiki/spaces/nxrtos/pages
56. https://www.st.com/resource/en/user_manual/um1718-stm32cubemx-for-stm32-configuration-and-initialization-c-code-generation-stmicroelectronics.pdf
57. https://www.hilscher.com/fileadmin/cms_upload/en-US/Resources/pdf/netX-90_Datasheet_10-2022-EN.pdf
58. https://www.digikey.no/en/maker/projects/getting-started-with-stm32-how-to-use-spi/09eab3dfe74c4d0391aaaa99b0a8ee17
59. https://deepbluembedded.com/stm32-gpio-registers-direct-access-fast-pin-control/
60. https://deepbluembedded.com/stm32-gpio-pin-read-lab-digital-input/
61. https://www.engineersgarage.com/accessing-ports-of-stm32-microcontroller/
62. https://embetronicx.com/tutorials/microcontrollers/stm32/stm32-gpio-tutorial/
63. https://www.youtube.com/watch?v=clegnIWCOBQ
64. https://www.waveshare.com/wiki/STM32CubeMX_Tutorial_Series:_GPIO
65. https://www.scribd.com/document/520582409/stm32f10x-gpio
66. https://deepbluembedded.com/stm32-gpio-write-pin-digital-output-lab/
67. https://www.st.com/resource/en/reference_manual/dm00224583-stm32f76xxx-and-stm32f77xxx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf
68. https://hilscher.atlassian.net/wiki/x/WACgB
69. https://wiki.stmicroelectronics.cn/stm32mcu/index.php?title=GPIO_feature_overview&oldid=8544
70. https://github.com/nitsky/stm32-example/blob/master/stm32/periph/src/stm32f30x_gpio.c
71. https://tp.prosoft.ru/docs/shared/webdav_bizproc_history_get/32210/32210/?ncc=1&force_download=1
72. https://community.st.com/t5/stm32-mcus-products/how-to-configure-gpio-interrupt-with-cubemx/td-p/469373
73. https://www.st.com/resource/en/reference_manual/rm0008-stm32f101xx-stm32f102xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

74. https://hilscher.atlassian.net/wiki/spaces/NETX/pages/144182900/Overview+netX+51+52+and+netX+90