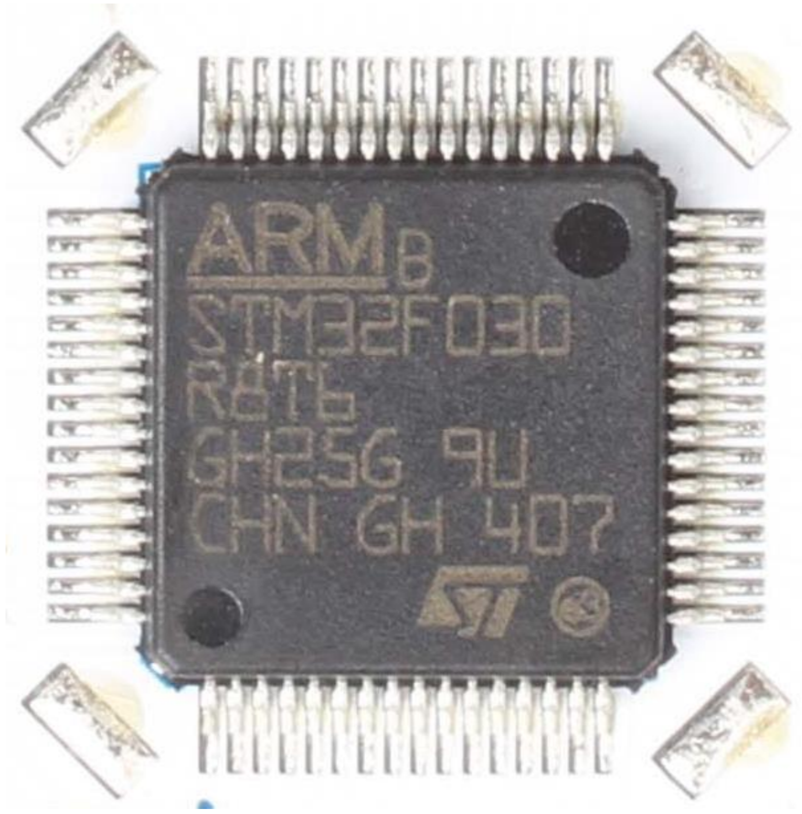
ARM Development Primer



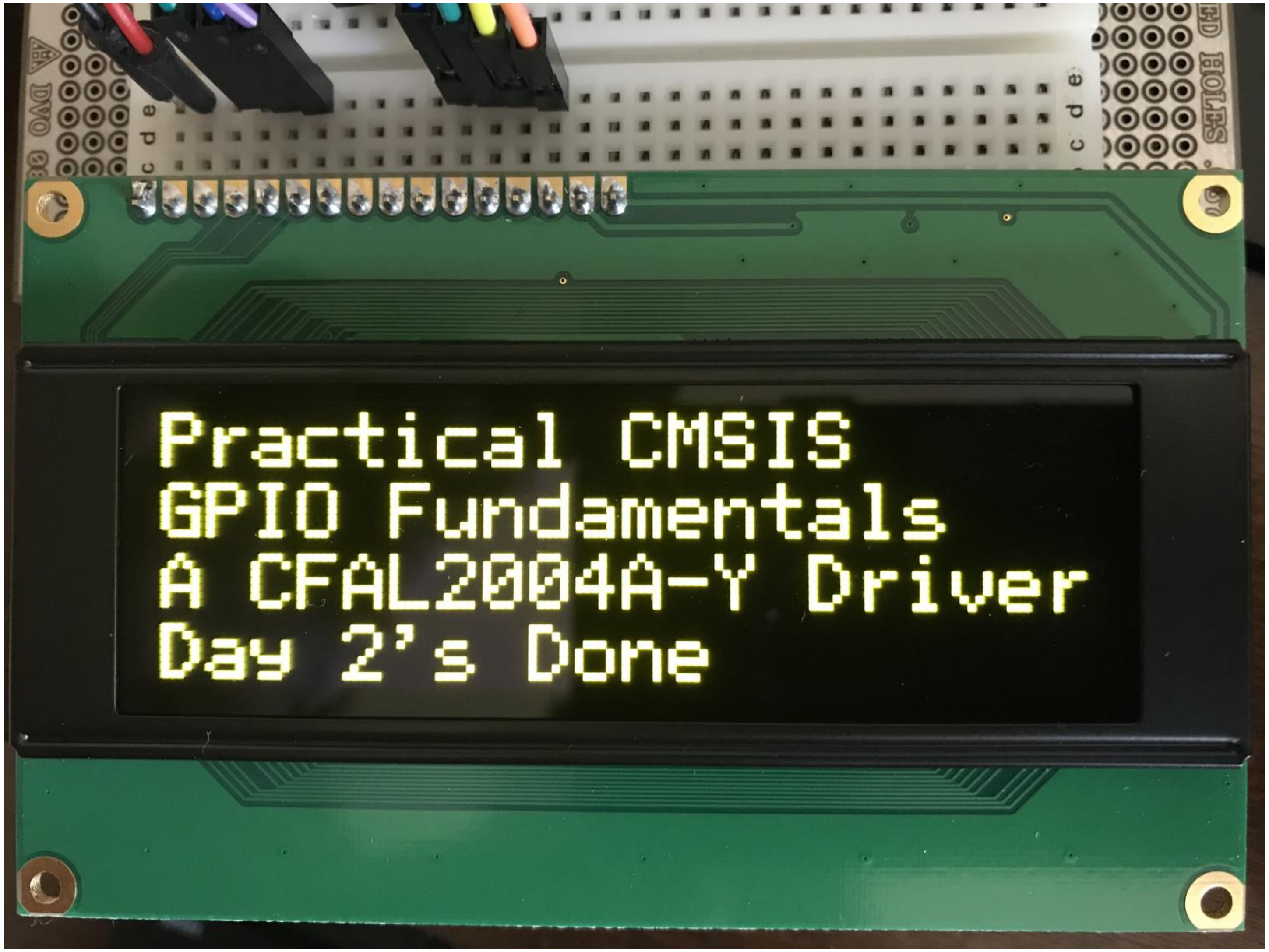
ARM I/O 101

November 14, 2017

FRED EADY

Presented by:

AGENDA



Practical CMSIS
GPIO Fundamentals
A CFAL2004A-Y Driver
Day 2's Done

Presented by:

ARM Development Primer

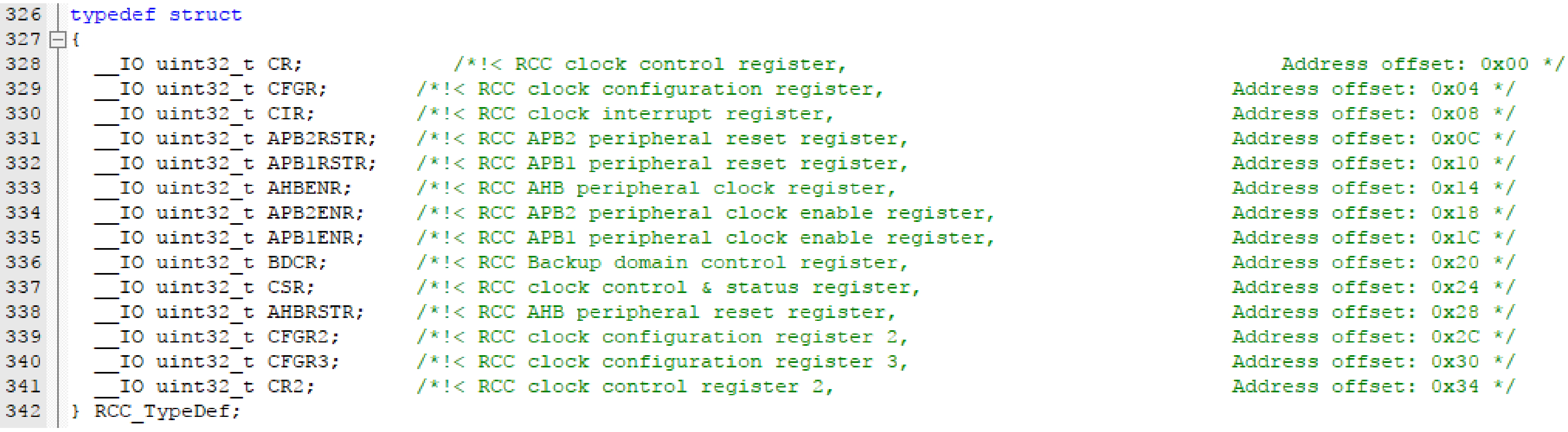
Practical CMSIS

```
********************************************************************************
* @file    stm32f030x8.h
* @author  MCD Application Team
* @version V2.2.3
* @date    29-January-2016
* @brief   CMSIS Cortex-M0 Device Peripheral Access Layer Header File.
*          This file contains all the peripheral register's definitions, bits
*          definitions and memory mapping for STM32F0xx devices.
*
*          This file contains:
*           - Data structures and the address mapping for all peripherals
*           - Peripheral's registers declarations and bits definition
*           - Macros to access peripheral’s registers hardware
*
********************************************************************************
```

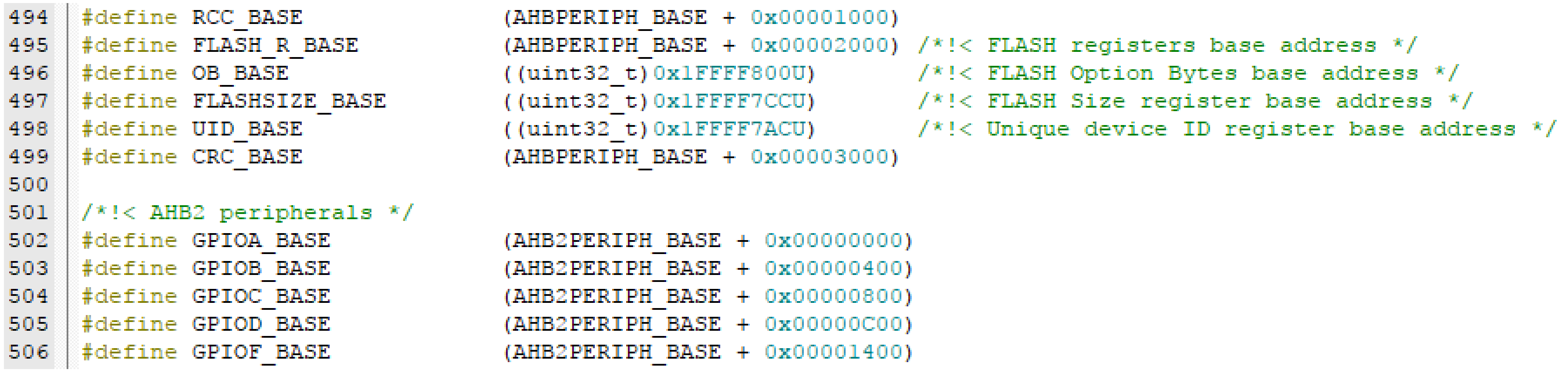
ARM Development Primer

Practical CMSIS

RCC – Reset and Clock Control



```c
typedef struct
{
  __IO uint32_t CR;            /*!< RCC clock control register,                    Address offset: 0x00 */
  __IO uint32_t CFGR;       /*!< RCC clock configuration register,            Address offset: 0x04 */
  __IO uint32_t CIR;        /*!< RCC clock interrupt register,                Address offset: 0x08 */
  __IO uint32_t APB2RSTR;   /*!< RCC APB2 peripheral reset register,          Address offset: 0x0C */
  __IO uint32_t APB1RSTR;   /*!< RCC APB1 peripheral reset register,          Address offset: 0x10 */
  __IO uint32_t AHBENR;     /*!< RCC AHB peripheral clock register,           Address offset: 0x14 */
  __IO uint32_t APB2ENR;    /*!< RCC APB2 peripheral clock enable register,   Address offset: 0x18 */
  __IO uint32_t APB1ENR;    /*!< RCC APB1 peripheral clock enable register,   Address offset: 0x1C */
  __IO uint32_t BDCR;       /*!< RCC Backup domain control register,          Address offset: 0x20 */
  __IO uint32_t CSR;        /*!< RCC clock control & status register,         Address offset: 0x24 */
  __IO uint32_t AHBRSTR;    /*!< RCC AHB peripheral reset register,           Address offset: 0x28 */
  __IO uint32_t CFGR2;      /*!< RCC clock configuration register 2,          Address offset: 0x2C */
  __IO uint32_t CFGR3;      /*!< RCC clock configuration register 3,          Address offset: 0x30 */
  __IO uint32_t CR2;        /*!< RCC clock control register 2,                Address offset: 0x34 */
} RCC_TypeDef;
```



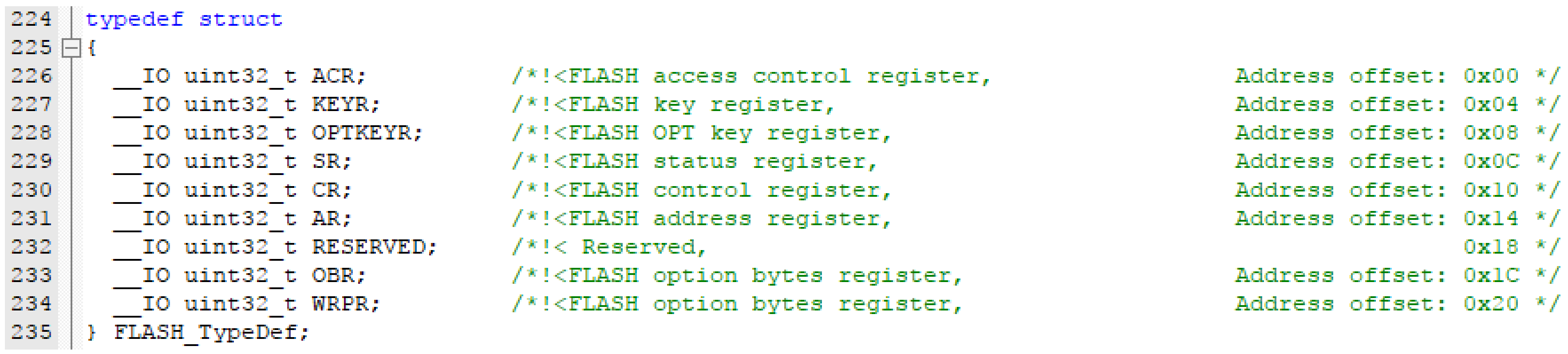
```c
#define RCC_BASE              (AHBPERIPH_BASE + 0x00001000)
#define FLASH_R_BASE          (AHBPERIPH_BASE + 0x00002000) /*!< FLASH registers base address */
#define OB_BASE               ((uint32_t)0x1FFFF800U)       /*!< FLASH Option Bytes base address */
#define FLASHSIZE_BASE        ((uint32_t)0x1FFFF7CCU)       /*!< FLASH Size register base address */
#define UID_BASE              ((uint32_t)0x1FFFF7ACU)       /*!< Unique device ID register base address */
#define CRC_BASE              (AHBPERIPH_BASE + 0x00003000)

/*!< AHB2 peripherals */
#define GPIOA_BASE            (AHB2PERIPH_BASE + 0x00000000)
#define GPIOB_BASE            (AHB2PERIPH_BASE + 0x00000400)
#define GPIOC_BASE            (AHB2PERIPH_BASE + 0x00000800)
#define GPIOD_BASE            (AHB2PERIPH_BASE + 0x00000C00)
#define GPIOF_BASE            (AHB2PERIPH_BASE + 0x00001400)
```

Presented by:

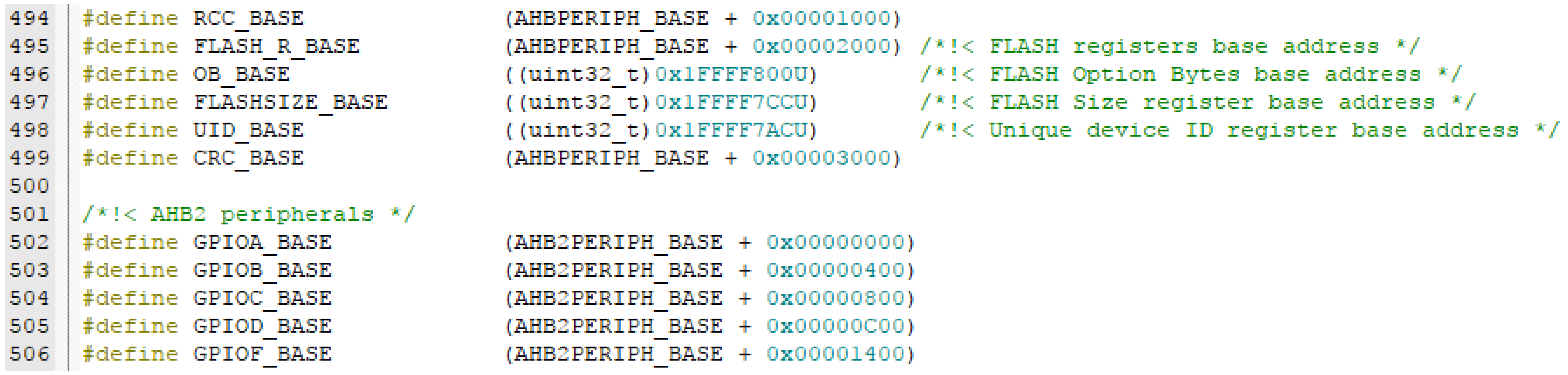
ARM Development Primer

Practical CMSIS

Flash



```c
typedef struct
{
  __IO uint32_t ACR;          /*!<FLASH access control register,                 Address offset: 0x00 */
  __IO uint32_t KEYR;         /*!<FLASH key register,                            Address offset: 0x04 */
  __IO uint32_t OPTKEYR;      /*!<FLASH OPT key register,                        Address offset: 0x08 */
  __IO uint32_t SR;           /*!<FLASH status register,                         Address offset: 0x0C */
  __IO uint32_t CR;           /*!<FLASH control register,                        Address offset: 0x10 */
  __IO uint32_t AR;           /*!<FLASH address register,                        Address offset: 0x14 */
  __IO uint32_t RESERVED;     /*!< Reserved,                                                     0x18 */
  __IO uint32_t OBR;          /*!<FLASH option bytes register,                   Address offset: 0x1C */
  __IO uint32_t WRPR;         /*!<FLASH option bytes register,                   Address offset: 0x20 */
} FLASH_TypeDef;
```



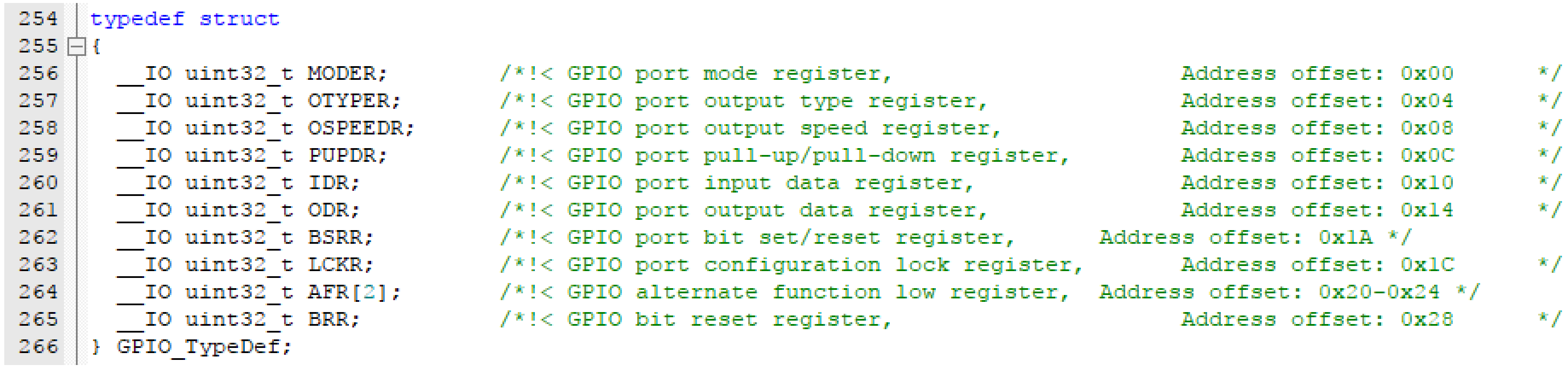
```c
#define RCC_BASE              (AHBPERIPH_BASE + 0x00001000)
#define FLASH_R_BASE          (AHBPERIPH_BASE + 0x00002000) /*!< FLASH registers base address */
#define OB_BASE               ((uint32_t)0x1FFFF800U)       /*!< FLASH Option Bytes base address */
#define FLASHSIZE_BASE        ((uint32_t)0x1FFFF7CCU)       /*!< FLASH Size register base address */
#define UID_BASE              ((uint32_t)0x1FFFF7ACU)       /*!< Unique device ID register base address */
#define CRC_BASE              (AHBPERIPH_BASE + 0x00003000)

/*!< AHB2 peripherals */
#define GPIOA_BASE            (AHB2PERIPH_BASE + 0x00000000)
#define GPIOB_BASE            (AHB2PERIPH_BASE + 0x00000400)
#define GPIOC_BASE            (AHB2PERIPH_BASE + 0x00000800)
#define GPIOD_BASE            (AHB2PERIPH_BASE + 0x00000C00)
#define GPIOF_BASE            (AHB2PERIPH_BASE + 0x00001400)
```

Presented by:

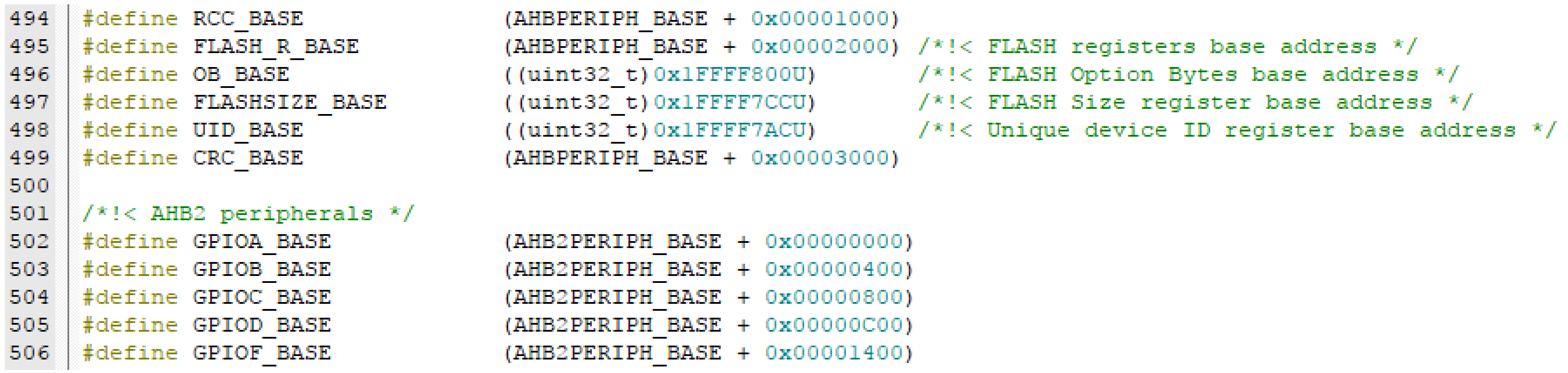
ARM Development Primer

Practical CMSIS

GPIO – General Purpose Input/Output



```c
typedef struct
{
  __IO uint32_t MODER;        /*!< GPIO port mode register,                    Address offset: 0x00      */
  __IO uint32_t OTYPER;       /*!< GPIO port output type register,             Address offset: 0x04      */
  __IO uint32_t OSPEEDR;      /*!< GPIO port output speed register,            Address offset: 0x08      */
  __IO uint32_t PUPDR;        /*!< GPIO port pull-up/pull-down register,       Address offset: 0x0C      */
  __IO uint32_t IDR;          /*!< GPIO port input data register,              Address offset: 0x10      */
  __IO uint32_t ODR;          /*!< GPIO port output data register,             Address offset: 0x14      */
  __IO uint32_t BSRR;         /*!< GPIO port bit set/reset register,      Address offset: 0x1A */
  __IO uint32_t LCKR;         /*!< GPIO port configuration lock register,      Address offset: 0x1C      */
  __IO uint32_t AFR[2];       /*!< GPIO alternate function low register,  Address offset: 0x20-0x24 */
  __IO uint32_t BRR;          /*!< GPIO bit reset register,                    Address offset: 0x28      */
} GPIO_TypeDef;
```



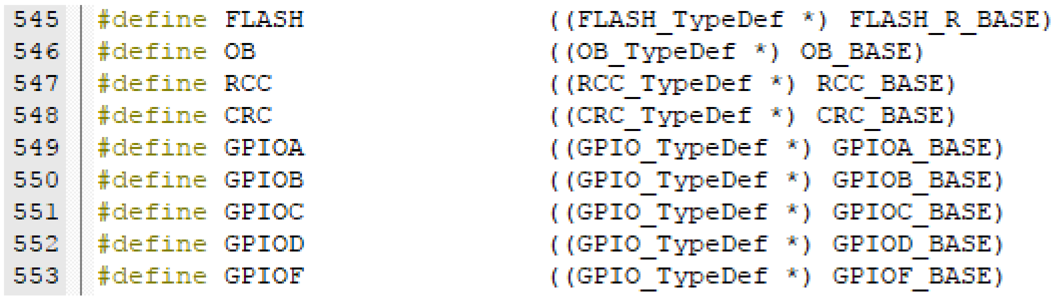
```c
#define RCC_BASE              (AHBPERIPH_BASE + 0x00001000)
#define FLASH_R_BASE          (AHBPERIPH_BASE + 0x00002000) /*!< FLASH registers base address */
#define OB_BASE               ((uint32_t)0x1FFFF800U)       /*!< FLASH Option Bytes base address */
#define FLASHSIZE_BASE        ((uint32_t)0x1FFFF7CCU)       /*!< FLASH Size register base address */
#define UID_BASE              ((uint32_t)0x1FFFF7ACU)       /*!< Unique device ID register base address */
#define CRC_BASE              (AHBPERIPH_BASE + 0x00003000)

/*!< AHB2 peripherals */
#define GPIOA_BASE            (AHB2PERIPH_BASE + 0x00000000)
#define GPIOB_BASE            (AHB2PERIPH_BASE + 0x00000400)
#define GPIOC_BASE            (AHB2PERIPH_BASE + 0x00000800)
#define GPIOD_BASE            (AHB2PERIPH_BASE + 0x00000C00)
#define GPIOF_BASE            (AHB2PERIPH_BASE + 0x00001400)
```

Presented by:

ARM Development Primer

Practical CMSIS

Peripheral Declarations



```
#define FLASH                   ((FLASH_TypeDef *) FLASH_R_BASE)
#define OB                      ((OB_TypeDef *) OB_BASE)
#define RCC                     ((RCC_TypeDef *) RCC_BASE)
#define CRC                     ((CRC_TypeDef *) CRC_BASE)
#define GPIOA                   ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB                   ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC                   ((GPIO_TypeDef *) GPIOC_BASE)
#define GPIOD                   ((GPIO_TypeDef *) GPIOD_BASE)
#define GPIOF                   ((GPIO_TypeDef *) GPIOF_BASE)
```

To Set I/O PIN PA.5 as OUTPUT-

Instead of: `(*GPIOA).MODER |= (GPIO_MODER_MODER5_0);`

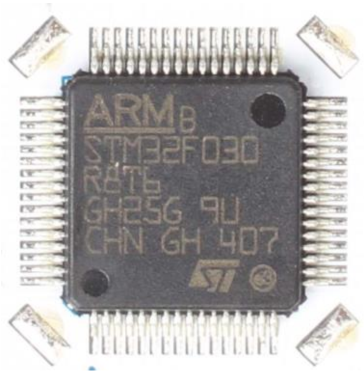
We use: `GPIOA->MODER |= (GPIO_MODER_MODER5_0);`

Presented by:

ARM Development Primer

GPIO Fundamentals

8.4.1 GPIO port mode register (GPIOx_MODER) (x =A..F)



Address offset:0x00

Reset values:

- 0x2800 0000 for port A
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODERy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O mode.

00: Input mode (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Presented by:

ARM Development Primer

GPIO Fundamentals

8.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..F)

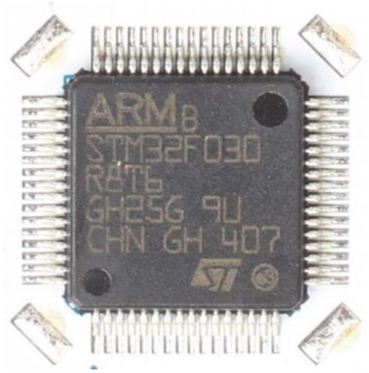
Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16	Reserved, must be kept at reset value.
Bits 15:0	**OTy:** Port x configuration bits (y = 0..15) These bits are written by software to configure the I/O output type. 0: Output push-pull (reset state) 1: Output open-drain

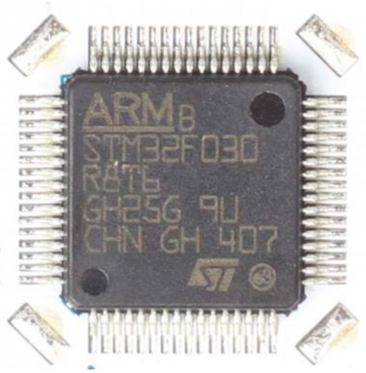


Presented by:

ARM Development Primer

GPIO Fundamentals

8.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..F)



Address offset: 0x0C

Reset values:

- 0x2400 0000 for port A
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **PUPDRy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O pull-up or pull-down

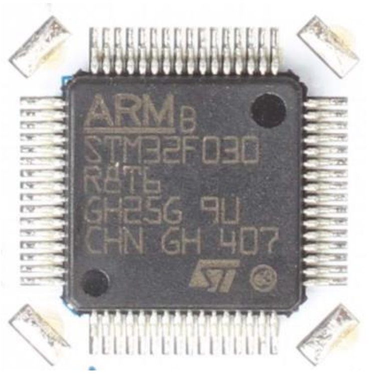
00: No pull-up, pull-down
01: Pull-up
10: Pull-down
11: Reserved

Presented by:

ARM Development Primer

GPIO Fundamentals

8.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..F)



Address offset: 0x08

Reset value:

- 0x0C00 0000 for port A
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OSPEEDR7 [1:0]		OSPEEDR6 [1:0]		OSPEEDR5 [1:0]		OSPEEDR4 [1:0]		OSPEEDR3 [1:0]		OSPEEDR2 [1:0]		OSPEEDR1 [1:0]		OSPEEDR0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **OSPEEDRy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

x0: Low speed
01: Medium speed
11: High speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

Presented by:

ARM Development Primer

GPIO Fundamentals

8.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A..F)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

8.4.11 GPIO port bit reset register (GPIOx_BRR) (x =A..F)

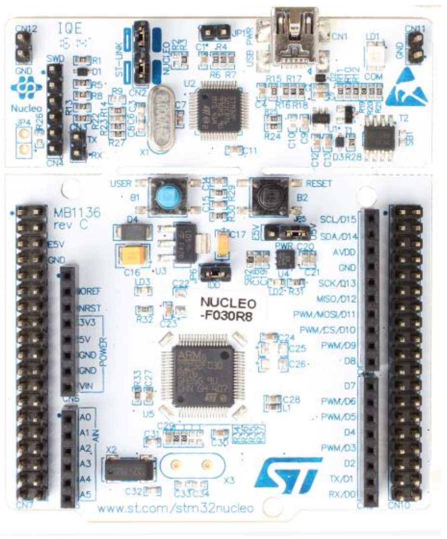
Address offset: 0x28
Reset value: 0x0000 0000

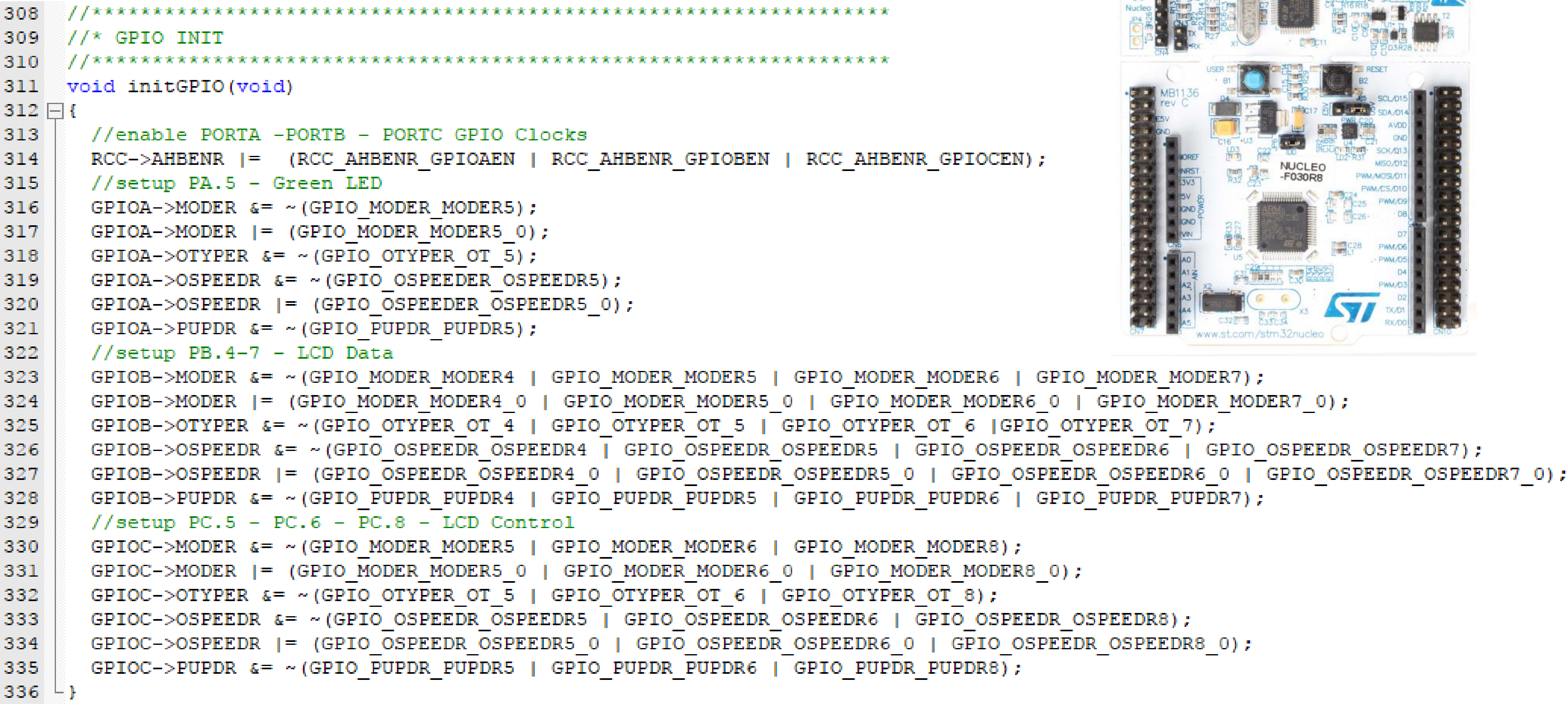
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

ARM Development Primer

GPIO Fundamentals



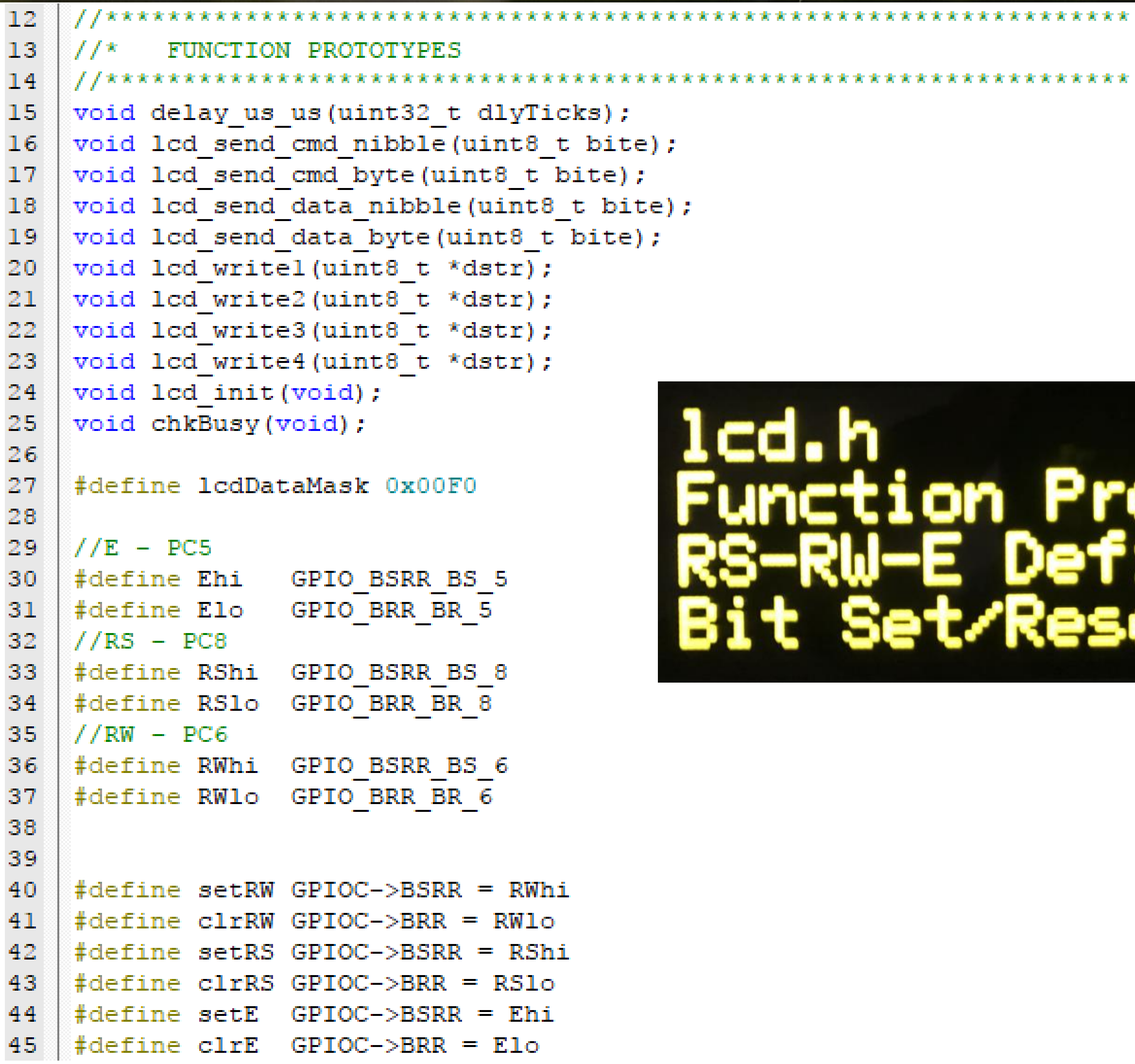


```
//*******************************************************************
//* GPIO INIT
//*******************************************************************
void initGPIO(void)
{
  //enable PORTA -PORTB - PORTC GPIO Clocks
  RCC->AHBENR |=  (RCC_AHBENR_GPIOAEN | RCC_AHBENR_GPIOBEN | RCC_AHBENR_GPIOCEN);
  //setup PA.5 - Green LED
  GPIOA->MODER &= ~(GPIO_MODER_MODER5);
  GPIOA->MODER |= (GPIO_MODER_MODER5_0);
  GPIOA->OTYPER &= ~(GPIO_OTYPER_OT_5);
  GPIOA->OSPEEDR &= ~(GPIO_OSPEEDER_OSPEEDR5);
  GPIOA->OSPEEDR |= (GPIO_OSPEEDER_OSPEEDR5_0);
  GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPDR5);
  //setup PB.4-7 - LCD Data
  GPIOB->MODER &= ~(GPIO_MODER_MODER4 | GPIO_MODER_MODER5 | GPIO_MODER_MODER6 | GPIO_MODER_MODER7);
  GPIOB->MODER |= (GPIO_MODER_MODER4_0 | GPIO_MODER_MODER5_0 | GPIO_MODER_MODER6_0 | GPIO_MODER_MODER7_0);
  GPIOB->OTYPER &= ~(GPIO_OTYPER_OT_4 | GPIO_OTYPER_OT_5 | GPIO_OTYPER_OT_6 |GPIO_OTYPER_OT_7);
  GPIOB->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEEDR4 | GPIO_OSPEEDR_OSPEEDR5 | GPIO_OSPEEDR_OSPEEDR6 | GPIO_OSPEEDR_OSPEEDR7);
  GPIOB->OSPEEDR |= (GPIO_OSPEEDR_OSPEEDR4_0 | GPIO_OSPEEDR_OSPEEDR5_0 | GPIO_OSPEEDR_OSPEEDR6_0 | GPIO_OSPEEDR_OSPEEDR7_0);
  GPIOB->PUPDR &= ~(GPIO_PUPDR_PUPDR4 | GPIO_PUPDR_PUPDR5 | GPIO_PUPDR_PUPDR6 | GPIO_PUPDR_PUPDR7);
  //setup PC.5 - PC.6 - PC.8 - LCD Control
  GPIOC->MODER &= ~(GPIO_MODER_MODER5 | GPIO_MODER_MODER6 | GPIO_MODER_MODER8);
  GPIOC->MODER |= (GPIO_MODER_MODER5_0 | GPIO_MODER_MODER6_0 | GPIO_MODER_MODER8_0);
  GPIOC->OTYPER &= ~(GPIO_OTYPER_OT_5 | GPIO_OTYPER_OT_6 | GPIO_OTYPER_OT_8);
  GPIOC->OSPEEDR &= ~(GPIO_OSPEEDR_OSPEEDR5 | GPIO_OSPEEDR_OSPEEDR6 | GPIO_OSPEEDR_OSPEEDR8);
  GPIOC->OSPEEDR |= (GPIO_OSPEEDR_OSPEEDR5_0 | GPIO_OSPEEDR_OSPEEDR6_0 | GPIO_OSPEEDR_OSPEEDR8_0);
  GPIOC->PUPDR &= ~(GPIO_PUPDR_PUPDR5 | GPIO_PUPDR_PUPDR6 | GPIO_PUPDR_PUPDR8);
}
```

Presented by:

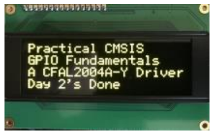
ARM Development Primer

A CFAL2004A-Y Driver



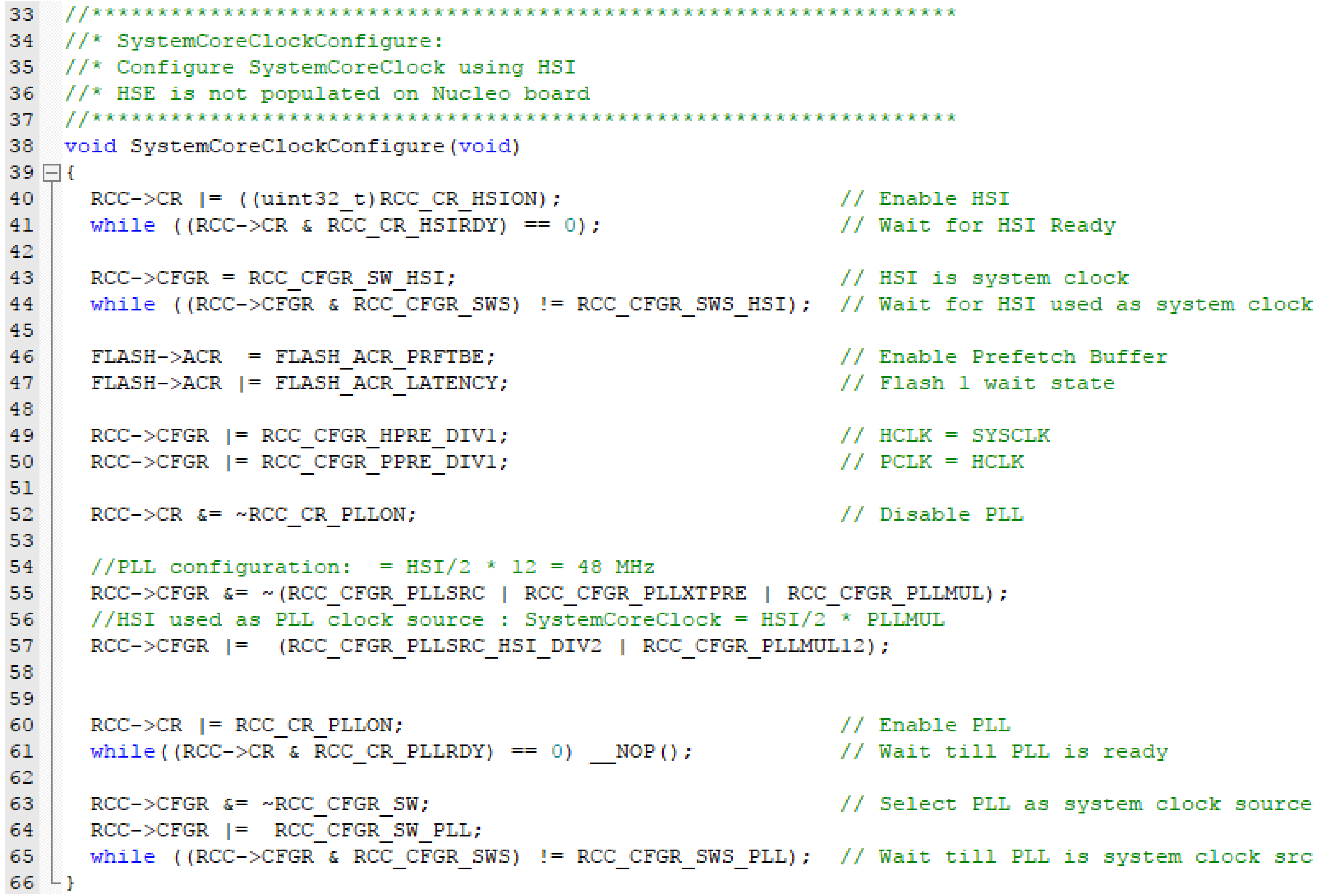
```
//*******************************************************************
//*   FUNCTION PROTOTYPES
//*******************************************************************
void delay_us_us(uint32_t dlyTicks);
void lcd_send_cmd_nibble(uint8_t bite);
void lcd_send_cmd_byte(uint8_t bite);
void lcd_send_data_nibble(uint8_t bite);
void lcd_send_data_byte(uint8_t bite);
void lcd_write1(uint8_t *dstr);
void lcd_write2(uint8_t *dstr);
void lcd_write3(uint8_t *dstr);
void lcd_write4(uint8_t *dstr);
void lcd_init(void);
void chkBusy(void);

#define lcdDataMask 0x00F0

//E - PC5
#define Ehi   GPIO_BSRR_BS_5
#define Elo   GPIO_BRR_BR_5
//RS - PC8
#define RShi  GPIO_BSRR_BS_8
#define RSlo  GPIO_BRR_BR_8
//RW - PC6
#define RWhi  GPIO_BSRR_BS_6
#define RWlo  GPIO_BRR_BR_6


#define setRW GPIOC->BSRR = RWhi
#define clrRW GPIOC->BRR = RWlo
#define setRS GPIOC->BSRR = RShi
#define clrRS GPIOC->BRR = RSlo
#define setE  GPIOC->BSRR = Ehi
#define clrE  GPIOC->BRR = Elo
```

lcd.h
Function Prototypes
RS-RW-E Definitions
Bit Set/Reset Macros



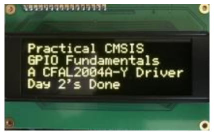
ARM Development Primer

A CFAL2004A-Y Driver



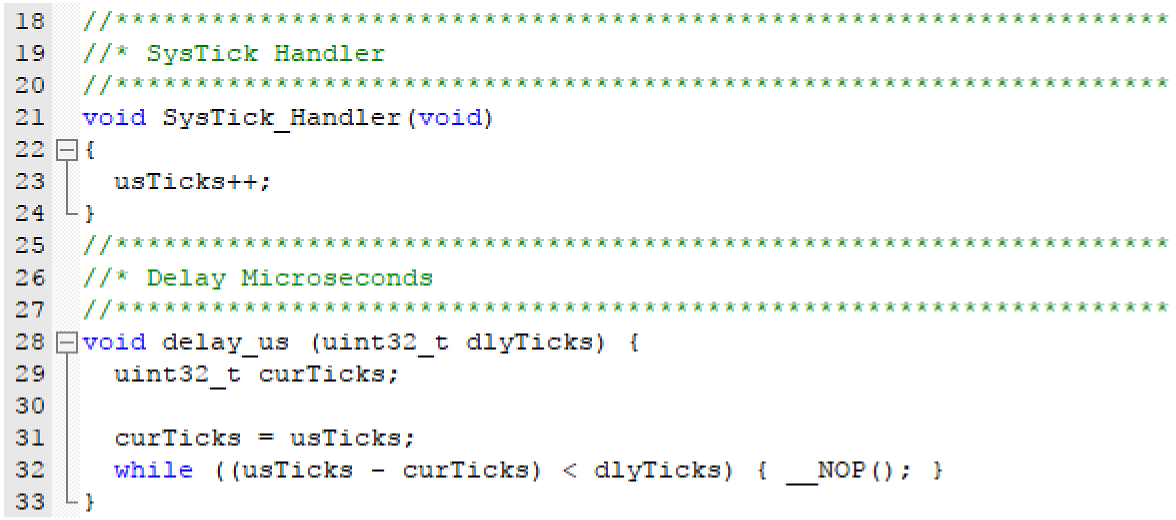
```
//*******************************************************************
//* SystemCoreClockConfigure:
//* Configure SystemCoreClock using HSI
//* HSE is not populated on Nucleo board
//*******************************************************************
void SystemCoreClockConfigure(void)
{
  RCC->CR |= ((uint32_t)RCC_CR_HSION);                     // Enable HSI
  while ((RCC->CR & RCC_CR_HSIRDY) == 0);                  // Wait for HSI Ready

  RCC->CFGR = RCC_CFGR_SW_HSI;                             // HSI is system clock
  while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_HSI);  // Wait for HSI used as system clock

  FLASH->ACR  = FLASH_ACR_PRFTBE;                          // Enable Prefetch Buffer
  FLASH->ACR |= FLASH_ACR_LATENCY;                         // Flash 1 wait state

  RCC->CFGR |= RCC_CFGR_HPRE_DIV1;                         // HCLK = SYSCLK
  RCC->CFGR |= RCC_CFGR_PPRE_DIV1;                         // PCLK = HCLK

  RCC->CR &= ~RCC_CR_PLLON;                                // Disable PLL

  //PLL configuration:  = HSI/2 * 12 = 48 MHz
  RCC->CFGR &= ~(RCC_CFGR_PLLSRC | RCC_CFGR_PLLXTPRE | RCC_CFGR_PLLMUL);
  //HSI used as PLL clock source : SystemCoreClock = HSI/2 * PLLMUL
  RCC->CFGR |=  (RCC_CFGR_PLLSRC_HSI_DIV2 | RCC_CFGR_PLLMUL12);


  RCC->CR |= RCC_CR_PLLON;                                 // Enable PLL
  while((RCC->CR & RCC_CR_PLLRDY) == 0) __NOP();           // Wait till PLL is ready

  RCC->CFGR &= ~RCC_CFGR_SW;                               // Select PLL as system clock source
  RCC->CFGR |=  RCC_CFGR_SW_PLL;
  while ((RCC->CFGR & RCC_CFGR_SWS) != RCC_CFGR_SWS_PLL);  // Wait till PLL is system clock src
}
```

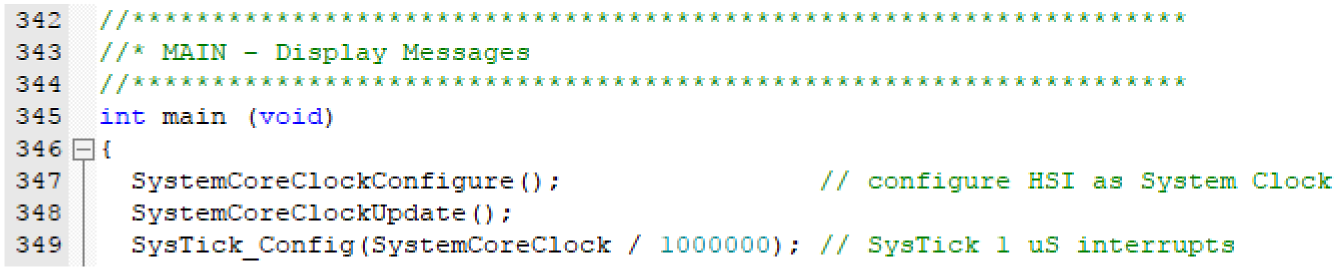
Presented by:



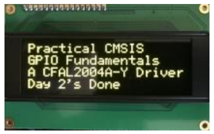
ARM Development Primer

A CFAL2004A-Y Driver


```
//*****************************************************************
//* SysTick Handler
//*****************************************************************
void SysTick_Handler(void)
{
  usTicks++;
}
//*****************************************************************
//* Delay Microseconds
//*****************************************************************
void delay_us (uint32_t dlyTicks) {
  uint32_t curTicks;

  curTicks = usTicks;
  while ((usTicks - curTicks) < dlyTicks) { __NOP(); }
}
```


```
//*****************************************************************
//* MAIN - Display Messages
//*****************************************************************
int main (void)
{
  SystemCoreClockConfigure();                    // configure HSI as System Clock
  SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock / 1000000); // SysTick 1 uS interrupts
```

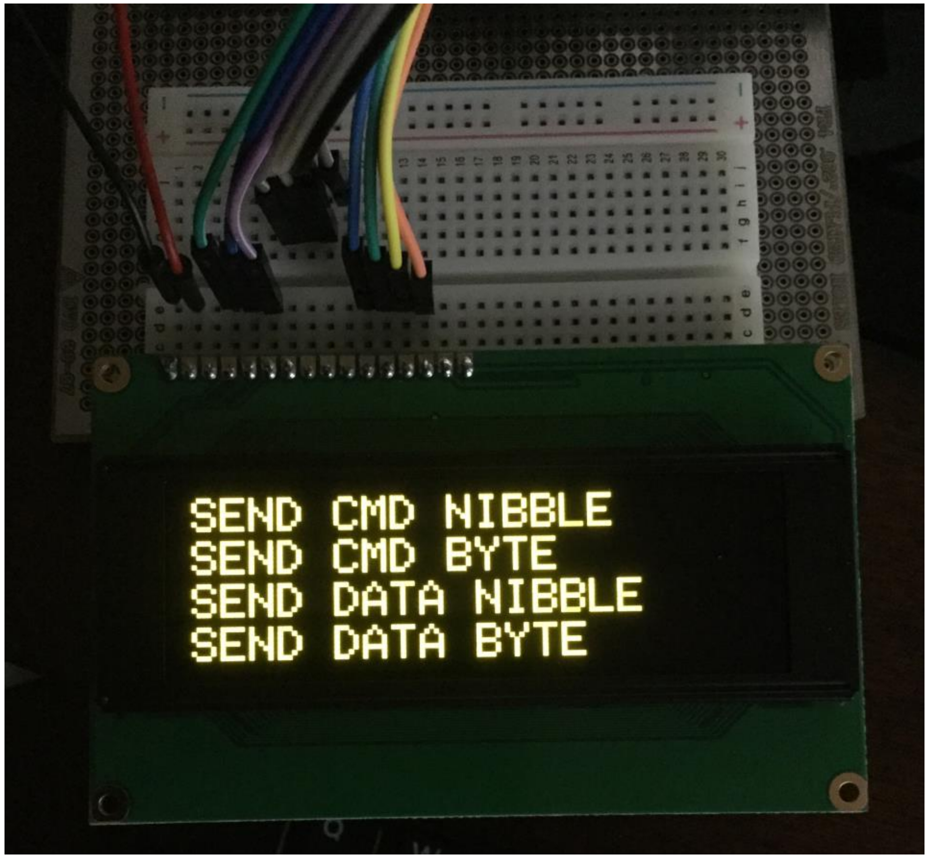
Presented by:



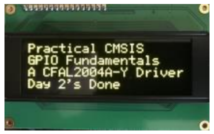
ARM Development Primer

A CFAL2004A-Y Driver

```
//**********************************************************************
//* SEND CMD NIBBLE
//**********************************************************************
void lcd_send_cmd_nibble(uint8_t bite)
{
    clrE;
    clrRS;
    GPIOB->ODR &= ~(lcdDataMask);
    GPIOB->ODR |= (bite & 0xF0);
    setE;
    delay_us(1);
    clrE;
}
//**********************************************************************
//* SEND CMD BYTE
//**********************************************************************
void lcd_send_cmd_byte(uint8_t bite)
{
  chkBusy();
  lcd_send_cmd_nibble(bite);
  lcd_send_cmd_nibble(bite << 4);
}
//**********************************************************************
//* SEND DATA NIBBLE
//**********************************************************************
void lcd_send_data_nibble(uint8_t bite)
{
    clrE;
    setRS;
    GPIOB->ODR &= ~(lcdDataMask);
    GPIOB->ODR |= (bite & 0xF0);
    setE;
    delay_us(1);
    clrE;
}
//**********************************************************************
//* SEND DATA BYTE
//**********************************************************************
void lcd_send_data_byte(uint8_t bite)
{
    chkBusy();
    lcd_send_data_nibble(bite);
    lcd_send_data_nibble(bite  << 4);
}
```

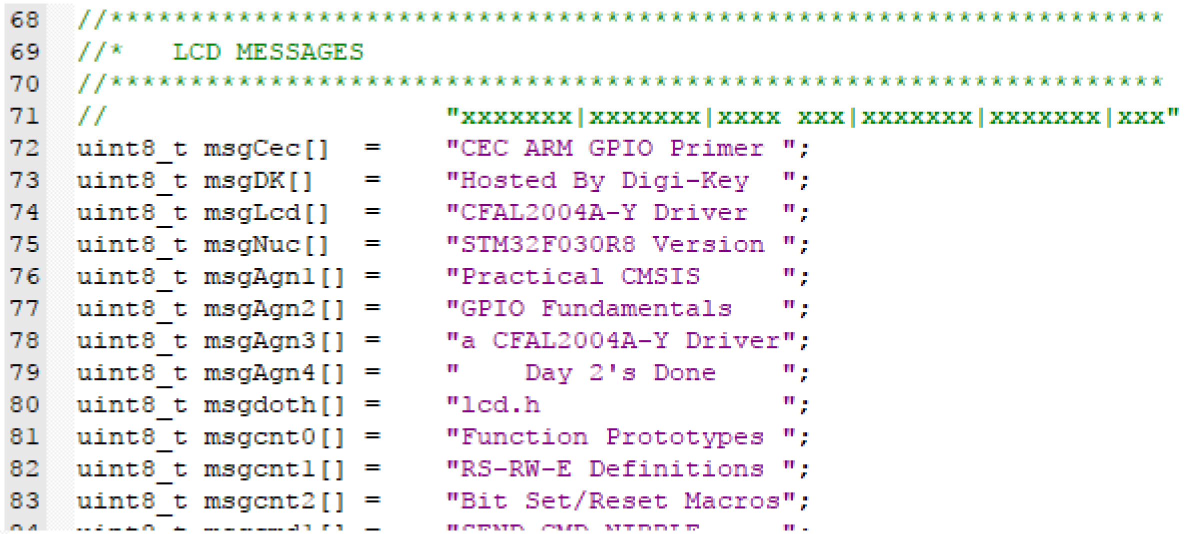


Presented by:

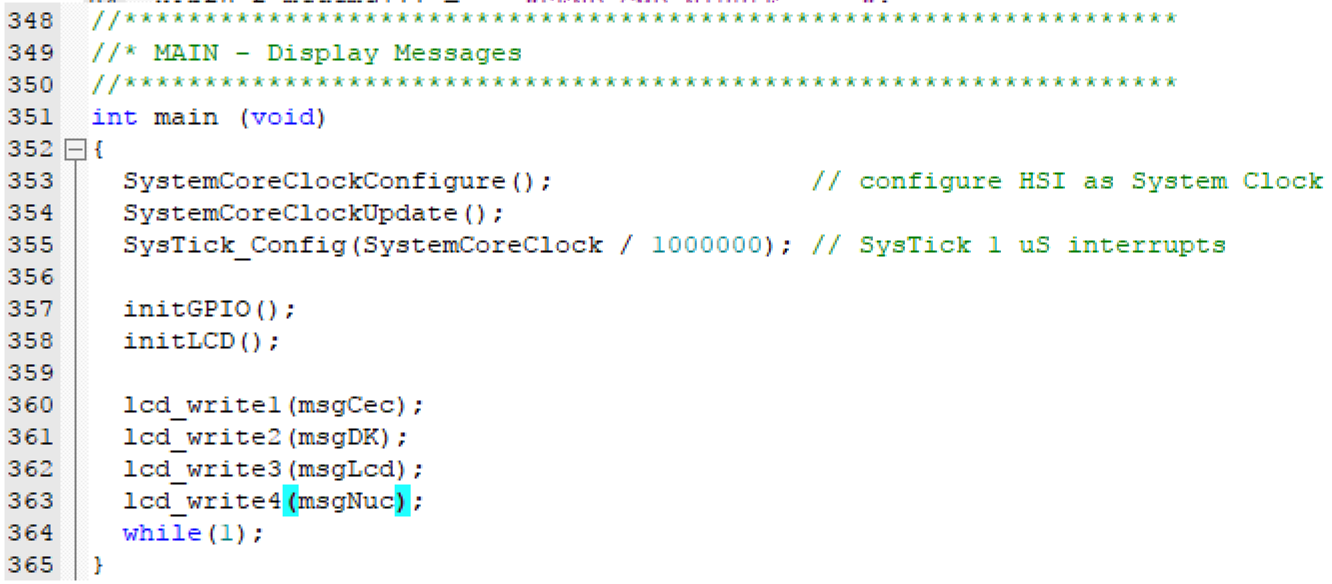


ARM Development Primer

A CFAL2004A-Y Driver



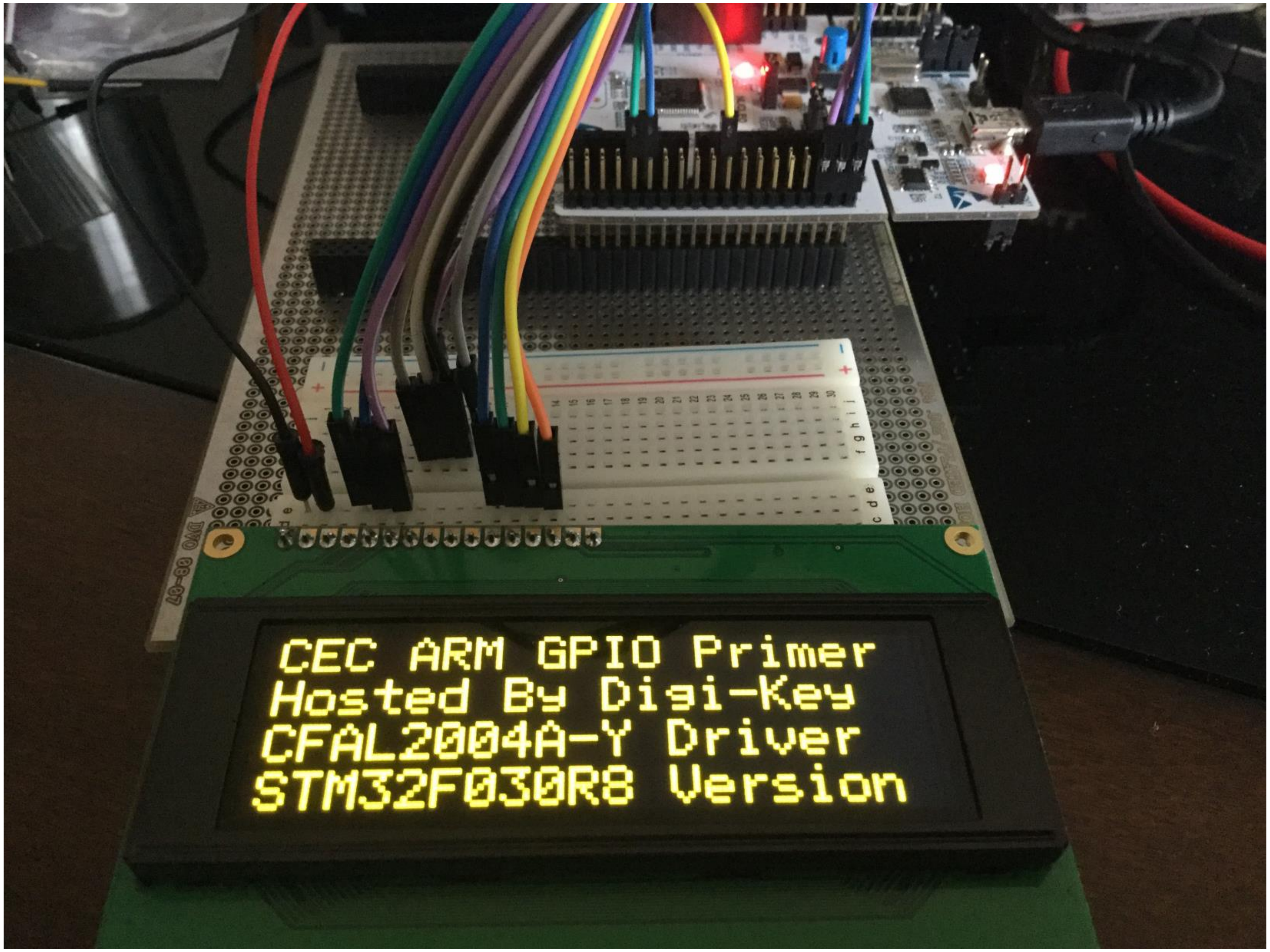
```
//**********************************************************************
//*    LCD MESSAGES
//**********************************************************************
//                     "xxxxxxx|xxxxxxx|xxxx xxx|xxxxxxx|xxxxxxx|xxx"
uint8_t msgCec[]  =    "CEC ARM GPIO Primer ";
uint8_t msgDK[]   =    "Hosted By Digi-Key  ";
uint8_t msgLcd[]  =    "CFAL2004A-Y Driver  ";
uint8_t msgNuc[]  =    "STM32F030R8 Version ";
uint8_t msgAgn1[] =    "Practical CMSIS     ";
uint8_t msgAgn2[] =    "GPIO Fundamentals   ";
uint8_t msgAgn3[] =    "a CFAL2004A-Y Driver";
uint8_t msgAgn4[] =    "     Day 2's Done   ";
uint8_t msgdoth[] =    "lcd.h               ";
uint8_t msgcnt0[] =    "Function Prototypes ";
uint8_t msgcnt1[] =    "RS-RW-E Definitions ";
uint8_t msgcnt2[] =    "Bit Set/Reset Macros";
```



```
//**********************************************************************
//* MAIN - Display Messages
//**********************************************************************
int main (void)
{
  SystemCoreClockConfigure();                     // configure HSI as System Clock
  SystemCoreClockUpdate();
  SysTick_Config(SystemCoreClock / 1000000); // SysTick 1 uS interrupts

  initGPIO();
  initLCD();

  lcd_write1(msgCec);
  lcd_write2(msgDK);
  lcd_write3(msgLcd);
  lcd_write4(msgNuc);
  while(1);
}
```

Presented by:

ARM Development Primer



Presented by:

ARM Development Primer

```
   Day 2's Done
We used CMSIS and
ARM GPIO to code
a CFAL2004A-Y Driver
```

Presented by: