Referenční příručka knihovny K8055 pro Arduino

*Technická dokumentace pro knihovnu **`K8055.h`** pro modul Arduino s rozhraním USB Host Shield 2.0 a experimentální desku Velleman K8055/K8055N*

Úvod a hardwarové propojení

Tato příručka popisuje softwarovou knihovnu `K8055.h` určenou pro komunikaci modulu Arduino s USB experimentální deskou Velleman K8055/K8055N). Hardwarové propojení je realizováno prostřednictvím rozšíření USB Host Shield 2.0 (*založeném na čipu MAX3421E*), které umožňuje Arduinu pracovat v režimu USB Host.

Hardwarová architektura a USB detekce

- **USB Host Shield:** Čip MAX3421E na USB Host Shieldu komunikuje s Arduinem přes sběrnici SPI a zajišťuje nízkoúrovňový přenos dat přes USB HID protokoly (využita knihovna `Usb.h`).
- **Deska Velleman K8055/K8055N:** Experimentální deska obsahuje 5 digitálních vstupů, 8 digitálních výstupů, 2 analogové vstupy (AD1, AD2), 2 analogové výstupy (DAC1, DAC2) a 2 hardwarové čítače pulzů.
- **USB Identifikace (VID/PID):** Knihovna identifikuje desku na základě konstanty `K8055_VENDOR_ID` (`0x10CF`). Product ID (PID) je dle zvolené adresy v rozsahu `0x5500` až `0x5503` (`K8055_PID_BASE` + adresa desky 0 až 3). Adresa karty je určena propojkami SK5/SK6 na desce.

Rozlišení revizí K8055 vs. K8055N

Při inicializaci knihovna odesílá testovací USB příkaz a vyhodnocuje přijatý bajt firmware:

- **K8055N (New Mode):** Pokud je deska identifikována jako novější revize K8055N, podporuje knihovna rozšířenou sadu příkazů, jako jsou 32-bitové čítače, hardwarové PWM výstupy a zpětné čtení stavu výstupů. Je nastavena i vyšší rychlost spolupráce s deskou.
- **K8055 (Legacy Mode):** V případě detekce starší varianty připojené USB desky knihovna pracuje v původním režimu K8055 s 16-bitovými čítači, standardními HID reporty a se sníženou rychlostí odezvy.

Časování a ochrana proti zahlcení sběrnice

Experimentální USB deska Velleman K8055 vyžaduje určitý minimální časový odstup mezi dvěma po sobě jdoucími příkazy. Knihovna pro tento účel obsahuje vestavěný časovač zajišťující tento časový odstup mezi zasílanými příkazy. Při detekci karty v metodě `task()` knihovna zkontroluje revizi firmware a dle toho nastaví automaticky toto časování.

- **Doporučené časování:** Výchozí interval je zvolen dle připojeného hardwaru. Pro desku K8055N je nastaveno výchozí časování na 10 ms, pro starší desky K8055 je interval navýšen na 20 ms.

Pokud některý z exemplářů USB desky Velleman K8055 vyžaduje hodnotu tohoto časování upravit, lze to programově provést ruční změnu pomocí metody `setCommandInterval()` – viz dále.

⚠ DŮLEŽITÉ UPOZORNĚNÍ K BEZPEČNOSTI A SPOLEHLIVOSTI

Komunikace mezi deskou Arduino s USB Host Shieldem a kartou Velleman K8055 zatím není 100% spolehlivá. Přestože knihovna obsahuje interní mechanismy kontroly odezvy a časování, může občas dojít k zamrznutí programu v důsledku neočekávaných stavů na USB sběrnici nebo zablokování komunikace.

Doporučené opatření: Při nasazení knihovny se důrazně doporučuje provozovat USB komunikaci v kombinaci s hardwarovým watchdogem (WDT) modulu Arduino, který v případě zaseknutí systému provede automatický restart modulu Arduino (viz ukázkový skript v závěru).

Omezení použití: Tato verze knihovny není určena pro řízení procesů v kritické infrastruktuře ani v aplikacích, kde by případné nestandardní chování, výpadek či zamrznutí řízení mohly způsobit věcné škody, zranění nebo ohrožení na životě.

1. Konstruktor, inicializace a správa připojení

K8055::K8055

```
K8055(USB *pHost);
```

Konstruktor třídy K8055. Přijímá ukazatel na instanci třídy `USB` (*USB Host Shield*). Inicializuje vnitřní stavové proměnné a výchozí časový interval mezi příkazy (20 ms).

Parametr / Návrat	Typ	Popis
pHost	USB*	Ukazatel na hlavní objekt USB sběrnice (*USB Host Shield*).

Ukázka použití:

```
USB Usb;
K8055 k8055(&Usb);
```

K8055::begin

```
bool begin();
```

Registruje HID report parser knihovny v objektu *HIDUniversal*. Metoda musí být zavolána ve funkci `setup()` po inicializaci USB sběrnice `Usb.Init()`.

Parametr / Návrat	Typ	Popis
Návratová hodnota	bool	Vrací `true`, pokud se registrace HID parseru zdařila, jinak `false`.

Ukázka použití:

```
if (Usb.Init() == -1) {
    Serial.println("OSC did not start");
}
if (!k8055.begin()) {
    Serial.println("Chyba registrace parseru!");
}
```

K8055::task

```
void task();
```

Zajišťuje periodické zpracování USB sběrnice a automatickou detekci a výchozí nastavení desky K8055 po jejím připojení do USB portu. Metoda musí být volána cyklicky v hlavní smyčce `loop()`.

Ukázka použití:

```
void loop() {
    k8055.task();
    if (k8055.connected()) {
        // Obsluha karty K8055
    }
}
```

K8055::connected

```
bool connected();
```

Vrací aktuální stav připojení desky Velleman K8055 k USB Host Shieldu.

Parametr / Návrat	Typ	Popis
Návratová hodnota	`bool`	Vrací `true`, pokud je karta připojena a úspěšně identifikována, jinak `false`.

Ukázka použití:

```
if (k8055.connected()) {
    // Provést operace na kartě
}
```

K8055::cardVersion

```
uint8_t cardVersion();
```

Vrací číslo revize karty.

Parametr / Návrat	Typ	Popis
Návratová hodnota	`uint8_t`	Vrací `1` pro původní K8055, `2` pro K8055N, nebo `0` pokud karta není připojena.

Ukázka použití:

```
uint8_t ver = k8055.cardVersion();
Serial.print("Revize karty: ");
Serial.println(ver);
```

K8055::getCardAddress

```
uint8_t getCardAddress();
```

Vrací hardwarovou adresu karty (`0` až `3`) – určeno propojkami SK5/SK6 na desce.

Parametr / Návrat	Typ	Popis
Návratová hodnota	`uint8_t`	Adresa karty v rozsahu `0` až `3`.

Ukázka použití:

```
Serial.print("Adresa karty: ");
Serial.println(k8055.getCardAddress());
```

K8055::isNewMode

```
bool isNewMode();
```

Zjistí, zda karta pracuje v novém režimu K8055N.

Parametr / Návrat	Typ	Popis
Návratová hodnota	`bool`	Vrací `true` pro verzi K8055N, `false` pro starší revizi K8055.

Ukázka použití:

```
if (k8055.isNewMode()) {
    Serial.println("Detekována karta K8055N s rozšířením PWM a 32-
bit čítači.");
}
```

K8055::setCommandInterval

```
void setCommandInterval(uint16_t intervalMs);
```

Nastavuje minimální pauzu mezi po sobě odesílanými USB příkazy. Chrání USB sběrnici a komunikátor desky před zahlcením.

Parametr / Návrat	Typ	Popis
`intervalMs`	`uint16_t`	Minimální odstup příkazů v milisekundách (výchozí 20 ms).

Poznámka:

Pokud není tento interval nastaven ručně, metoda task() jej zvolí automaticky podle firmwaru (10 ms pro novější K8055N, 20 ms pro standardní K8055). Ruční nastavení hodnoty prodlevy pomocí metody `setCommandInterval()` (*např. při výchozím nastavení v* `setup()`) má přednost před automatickou detekcí v metodě `task()`.

Ukázka použití:

```
k8055.setCommandInterval(25);
```

K8055::getCommandInterval

```
uint16_t getCommandInterval();
```

Vrací aktuálně nastavený minimální interval mezi USB příkazy.

Parametr / Návrat	Typ	Popis
Návratová hodnota	uint16_t	Interval v ms.

Ukázka použití:

```
Serial.print("Min. interval: ");
Serial.println(k8055.getCommandInterval());
```

K8055:: resetConnection

```
void resetConnection();
```

Ručně vynutí opětovnou detekci a inicializaci karty na USB sběrnici.

Ukázka použití:

```
k8055.resetConnection();
```

2. Řízení digitálních výstupů (OUT1–OUT8)

K8055::writeAllDigital

```
void writeAllDigital(uint8_t data);
```

Zapíše pomocí bitové masky stav na všech 8 digitálních výstupů současně.

Parametr / Návrat	Typ	Popis
data	uint8_t	Bitová maska výstupů (Bit 0 = OUT1 až Bit 7 = OUT8). Hodnota 1 = ZAP, 0 = VYP.

Ukázka použití:

```
k8055.writeAllDigital(0b10101010); // Zapne výstupy
OUT2,OUT4,OUT6,OUT8
```

K8055::setDigitalChannel

```
void setDigitalChannel(uint8_t channel);
```

Aktivuje (nastaví na LOG. 1) jeden konkrétní digitální výstupní kanál.

Parametr / Návrat	Typ	Popis
channel	uint8_t	Číslo kanálu výstupu v rozsahu 1 až 8.

Ukázka použití:

```
k8055.setDigitalChannel(3); // Zapne výstup OUT3
```

K8055::clearDigitalChannel

```
void clearDigitalChannel(uint8_t channel);
```

Deaktivuje (nastaví na LOG. 0) jeden konkrétní digitální výstupní kanál.

Parametr / Návrat	Typ	Popis
channel	uint8_t	Číslo kanálu výstupu v rozsahu 1 až 8.

Ukázka použití:

```
k8055.clearDigitalChannel(3); // Vypne výstup OUT3
```

K8055::setAllDigital

```
void setAllDigital();
```

Nastaví všechny digitální výstupy OUT1 až OUT8 do stavu LOG. 1.

Ukázka použití:

```
k8055.setAllDigital(); // Zapne všechny digitální výstupy
```

K8055::clearAllDigital

```
void clearAllDigital();
```

Vypne všechny digitální výstupy `OUT1` až `OUT8` (stavy LOG. 0).

Ukázka použití:

```
k8055.clearAllDigital(); // Vypne všechny digitální výstupy
```

K8055::readBackDigitalOut

```
uint8_t readBackDigitalOut();
```

Přečte aktuální stav digitálních výstupů. U K8055N vyžádá stav přímo z karty, u K8055 vrací uložený interní stav.

Parametr / Návrat	Typ	Popis
Návratová hodnota	`uint8_t`	Bitová maska aktuálních digitálních výstupů (`0x00` až `0xFF`).

Ukázka použití:

```
uint8_t st = k8055.readBackDigitalOut();
Serial.print("Stav výstupů: ");
Serial.println(st, BIN);
```

3. Řízení analogových a PWM výstupů (DAC1, DAC2, PWM)

K8055::outputAnalogChannel

```
void outputAnalogChannel(uint8_t channel, uint8_t data);
```

Nastaví analogovou hodnotu na výstupním kanálu `DAC1` nebo `DAC2`.

Parametr / Návrat	Typ	Popis
`channel`	`uint8_t`	Číslo analogového kanálu (`1` = `DAC1`, `2` = `DAC2`).
`data`	`uint8_t`	Výstupní hodnota v rozsahu `0` až `255` (`0` = 0 V, `255` = U_{max}).

Ukázka použití:

```
k8055.outputAnalogChannel(1, 128); // Nastaví DAC1 na 50 % rozsahu
```

K8055::outputAllAnalog

```
void outputAllAnalog(uint8_t data1, uint8_t data2);
```

Nastaví výstupní hodnotu na obou analogových kanálech `DAC1` a `DAC2` v jediném USB příkazu.

Parametr / Návrat	Typ	Popis
data1	uint8_t	Hodnota pro `DAC1` (`0` až `255`).
data2	uint8_t	Hodnota pro `DAC2` (`0` až `255`).

Ukázka použití:

```
k8055.outputAllAnalog(255, 64); // DAC1 na 100 %, DAC2 na 25 %
```

K8055::clearAnalogChannel

```
void clearAnalogChannel(uint8_t channel);
```

Vynuluje zvolený analogový kanál (nastaví výstup na `0`).

Parametr / Návrat	Typ	Popis
channel	uint8_t	Číslo kanálu (`1` = `DAC1`, `2` = `DAC2`).

Ukázka použití:

```
k8055.clearAnalogChannel(1);
```

K8055::clearAllAnalog

```
void clearAllAnalog();
```

Vynuluje oba analogové výstupy `DAC1` i `DAC2` (nastaví na `0`, `0`).

Ukázka použití:

```
k8055.clearAllAnalog();
```

K8055::setAnalogChannel

```
void setAnalogChannel(uint8_t channel);
```

Nastaví zvolený analogový výstup na maximální hodnotu (`255`).

Parametr / Návrat	Typ	Popis
channel	uint8_t	Číslo kanálu (1 = DAC1, 2 = DAC2).

Ukázka použití:

```
k8055.setAnalogChannel(2);
```

K8055::setAllAnalog

```
void setAllAnalog();
```

Nastaví oba analogové výstupy DAC1 i DAC2 na maximální hodnotu (255, 255).

Ukázka použití:

```
k8055.setAllAnalog();
```

K8055::readBackAnalogOut

```
void readBackAnalogOut(uint8_t &data1, uint8_t &data2);
```

Přečte aktuálně nastavené hodnoty analogových výstupů. U K8055N dochází k přečtení z hardwaru karty, u K8055 se vrací uložený interní stav.

Parametr / Návrat	Typ	Popis
data1	uint8_t&	Odkaz na proměnnou pro uložení hodnoty DAC1.
data2	uint8_t&	Odkaz na proměnnou pro uložení hodnoty DAC2.

Ukázka použití:

```
uint8_t a1, a2;
k8055.readBackAnalogOut(a1, a2);
Serial.print("DAC1: ");
Serial.println(a1);
```

K8055::setPWM

```
void setPWM(uint8_t channel, uint8_t data, uint8_t frequency);
```

Konfiguruje hardwarový PWM (pulzně modulovaný) výstup – dostupné výhradně na kartách K8055N, u starších K8055 se neprovádí žádná akce.

Parametr / Návrat	Typ	Popis
channel	uint8_t	Číslo PWM kanálu (1 nebo 2).
data	uint8_t	Střída PWM v rozsahu 0 až 255.
frequency	uint8_t	Frekvenční režim PWM v rozsahu 1 až 3 (1 = 23,43 kHz, 2 = 2,93 kHz, 3 = 0,37 kHz).

Ukázka použití:

```
if (k8055.isNewMode()) {
    k8055.setPWM(1, 128, 2); // kanál 1, 50% střída, freq. 2,93 KHz
}
```

4. Čtení digitálních a analogových vstupů (IN1–IN5, AD1, AD2)

K8055::readDigitalChannel

```
bool readDigitalChannel(uint8_t channel);
```

Přečte stav jednoho konkrétního digitálního vstupu IN1 až IN5.

Parametr / Návrat	Typ	Popis
channel	uint8_t	Číslo vstupu v rozsahu 1 až 5.
Návratová hodnota	bool	Vrací true pokud je vstup aktivní (LOG. 1), jinak false.

Ukázka použití:

```
if (k8055.readDigitalChannel(1)) {
    Serial.println("Tlačítko na IN1 stisknuto!");
}
```

K8055::readAllDigital

```
uint8_t readAllDigital();
```

Přečte stav všech pěti digitálních vstupů naráz.

Parametr / Návrat	Typ	Popis
Návratová hodnota	uint8_t	Bitová maska digitálních vstupů (Bit 0 = IN1, Bit 1 = IN2, Bit 2 = IN3, Bit 3 = IN4, Bit 4 = IN5).

Ukázka použití:

```
uint8_t inputs = k8055.readAllDigital();
if (inputs & 0x01) {
    /* IN1 aktivní */
}
```

K8055::readAnalogChannel

```
uint8_t readAnalogChannel(uint8_t channel);
```

Přečte napěťovou hodnotu z analogového vstupu `AD1` nebo `AD2`.

Parametr / Návrat	Typ	Popis
channel	uint8_t	Číslo analogového vstupu (`1` = `AD1`, `2` = `AD2`).
Návratová hodnota	uint8_t	8-bitová hodnota analogově-digitálního převodníku (`0` až `255`).

Ukázka použití:

```
uint8_t valAD1 = k8055.readAnalogChannel(1);
Serial.print("AD1: ");
Serial.println(valAD1);
```

K8055::readAllAnalog

```
void readAllAnalog(uint8_t &data1, uint8_t &data2);
```

Přečte hodnoty obou analogových vstupů `AD1` a `AD2` v jediném dotazu.

Parametr / Návrat	Typ	Popis
data1	uint8_t&	Odkaz na proměnnou pro uložení hodnoty `AD1` (`0`–`255`).
data2	uint8_t&	Odkaz na proměnnou pro uložení hodnoty `AD2` (`0`–`255`).

Ukázka použití:

```
uint8_t ad1, ad2;
k8055.readAllAnalog(ad1, ad2);
Serial.print("AD1=");
Serial.print(ad1);
Serial.print(" AD2=");
Serial.println(ad2);
```

5. Správa hardwarových čítačů (`Counter 1` & `Counter 2`)

K8055::readCounter

```
uint32_t readCounter(uint8_t counterNro);
```

Přečte plnou hodnotu impulzního čítače `1` nebo `2`. U karty K8055N vrací 32-bitovou hodnotu, u starší K8055 vrací 16-bitovou hodnotu.

Parametr / Návrat	Typ	Popis
counterNro	uint8_t	Číslo čítače (`1` = `Counter 1` na `IN1`, `2` = `Counter 2` na `IN2`).
Návratová hodnota	uint32_t	Počet načítaných impulzů (32-bit nebo 16-bit – *dle revize karty*).

Ukázka použití:

```
uint32_t cnt1 = k8055.readCounter(1);
Serial.print("Impulzy Čítač 1: ");
Serial.println(cnt1);
```

K8055::readCounter16

```
uint16_t readCounter16(uint8_t counterNro);
```

Přečte spodních 16 bitů hodnoty čítače 1 nebo 2.

Parametr / Návrat	Typ	Popis
counterNro	uint8_t	Číslo čítače (`1` nebo `2`).
Návratová hodnota	uint16_t	16-bitová hodnota čítače (`0` až `65535`).

Ukázka použití:

```
uint16_t cnt16 = k8055.readCounter16(2);
```

K8055::resetCounter

```
void resetCounter(uint8_t counterNro);
```

Vynuluje stav zvoleného hardwarového čítače (`Counter 1` nebo `Counter 2`).

Parametr / Návrat	Typ	Popis
counterNro	uint8_t	Číslo čítače k vynulování (`1` nebo `2`).

Ukázka použití:

```
k8055.resetCounter(1); // Vynuluje Čítač 1
```

K8055::setCounterDebounceTime

```
void setCounterDebounceTime(uint8_t counterNro,
                            uint16_t debounceTime);
```

Nastavuje v milisekundách odrušovací čas (*debounce time*) pro digitální vstupy čítačů.

Parametr / Návrat	Typ	Popis
counterNro	uint8_t	Číslo čítače (1 nebo 2).
debounceTime	uint16_t	Odrušovací čas v milisekundách (např. 2 =2 ms, 10 = 10 ms atd.).

Ukázka použití:

```
k8055.setCounterDebounceTime(1, 10); // Odrušení 10 ms na Čítači 1
```

6. Kompletní čtení stavu

K8055::readAll

```
void readAll(uint8_t &digIn, uint8_t &an1, uint8_t &an2,
             uint32_t &cnt1, uint32_t &cnt2);
```

Nejefektivnější metoda pro kompletní přečtení všech vstupů karty v jediném vyžádání. U K8055N se v jediném paketu vrátí digitální vstupy, oba analogové vstupy a 16-bitové hodnoty obou čítačů.

Parametr / Návrat	Typ	Popis
digIn	uint8_t&	Proměnná pro stav digitálních vstupů IN1–IN5.
an1	uint8_t&	Proměnná pro uložení hodnoty AD1.
an2	uint8_t&	Proměnná pro uložení hodnoty AD2.
cnt1	uint32_t&	Proměnná pro uložení hodnota čítače 1.
cnt2	uint32_t&	Proměnná pro uložení hodnota čítače 2.

Ukázka použití:

```
uint8_t dIn, a1, a2;
uint32_t c1, c2;
k8055.readAll(dIn, a1, a2, c1, c2);
Serial.print("IN:");
Serial.print(dIn, BIN);
Serial.print(" AD1:");
Serial.print(a1);
Serial.print(" C1:");
Serial.println(c1);
```

7. Ukázkový skript pro kompletní diagnostiku komunikace Arduino+USB Host Shield ↔ USB karta Velleman K8055(N)

Následující skript testuje funkce knihovny K8055 – od inicializace USB Host Shieldu, přes detekci karty K8055/K8055N, periodické čtení všech analogových a digitálních vstupů/výstupů i čítačů a sekvenční řízení výstupů.

Skript: `K8055_Example.ino`

Tento kompletní ukázkový skript je připraven k okamžitému použití a je přímo součástí distribučního ZIP balíčku knihovny K8055. Kód tak není třeba opisovat z té příručky, stačí jej otevřít přímo v prostředí Arduino IDE. Po nainstalování knihovny do prostředí Arduino IDE (**Projekt** → **Zahrnout knihovnu** → **Přidat .ZIP knihovnu…**) naleznete skript přímo v hlavní nabídce: **Soubor** → **Příklady** → **K8055** → **K8055_Demo**.

```
/*
 * K8055 Library Complete Function Tester for Arduino UNO + USB Host Shield
 */

#include <Usb.h>
#include <K8055.h>

#include <avr/wdt.h>

USB Usb;
K8055 k8055(&Usb);

enum TestState {
    STATE_WAIT_CONNECT,
    STATE_CARD_INFO,
    STATE_TEST_DIGITAL_OUT,
    STATE_TEST_ANALOG_OUT,
    STATE_TEST_READBACK,
    STATE_TEST_DIGITAL_IN,
    STATE_TEST_ANALOG_IN,
    STATE_TEST_COUNTERS,
    STATE_TEST_READ_ALL,
```

```
    STATE_TEST_COMPLETE,
    STATE_TEST_END
};

TestState currentState = STATE_WAIT_CONNECT;

unsigned long stateTimer = 0;
unsigned long printTimer = 0;
uint8_t subStep = 0;

// Pomocná funkce pro obnovení stavu desky platným příkazem WriteAllOutputs
(cmd 5)
void refreshK8055Board() {
    uint8_t currentDig = k8055.readBackDigitalOut();
    k8055.writeAllDigital(currentDig);
}

void setup() {
    Serial.begin(115200);
    while (!Serial);

    Serial.println(F("========================================="));
    Serial.println(F("  Arduino UNO K8055 Complete Test Program"));
    Serial.println(F("========================================="));

    if (Usb.Init() == -1) {
        Serial.println(F("[CHYBA] USB Host Shield se nepodařilo
inicializovat!"));
        while (1);
    }

    if (!k8055.begin()) {
        Serial.println(F("[CHYBA] Nepodařilo se inicializovat K8055
parser!"));
        while (1);
    }

    Serial.println(F("[OK] USB Host i K8055 inicializovány. Čekám na
připojení desky..."));

    wdt_enable(WDTO_2S);
}

void loop() {
    // 1. Neustálá obsluha USB sběrnice bez blokování delay()
    k8055.task();

    // 2. Kontrola připojení
    if (!k8055.connected() && currentState != STATE_WAIT_CONNECT) {
        Serial.println(F("\n[VAROVÁNÍ] K8055 deska odpojena! Čekám na
opětovné připojení..."));
        currentState = STATE_WAIT_CONNECT;
        return;
    }

    // 3. Stavový automat
    switch (currentState) {

        case STATE_WAIT_CONNECT:
```

```
            if (k8055.connected()) {
                Serial.println(F("\n[STATUS] K8055 deska připojena!"));
                currentState = STATE_CARD_INFO;
            }
            break;

        case STATE_CARD_INFO:
            Serial.println(F("\n--- 1. INFORMACE O DESCE ---"));
            Serial.print(F("Adresa desky (Jumpery SK5/SK6): "));
            Serial.println(k8055.getCardAddress());

            Serial.print(F("Verze firmware K8055:   v"));
            Serial.println(k8055.cardVersion());

            subStep = 0;
            stateTimer = millis();
            currentState = STATE_TEST_DIGITAL_OUT;
            Serial.println(F("\n--- 2. TEST DIGITÁLNÍCH VÝSTUPŮ (OUT1 -
OUT8) ---"));
            break;

        case STATE_TEST_DIGITAL_OUT:
            if (millis() - stateTimer >= 300) {
                stateTimer = millis();

                if (subStep < 8) {
                    uint8_t ch = subStep + 1;
                    Serial.print(F("Zapínám OUT"));
                    Serial.println(ch);
                    k8055.setDigitalChannel(ch);
                    subStep++;
                } else if (subStep < 16) {
                    uint8_t ch = (subStep - 8) + 1;
                    Serial.print(F("Vypínám OUT"));
                    Serial.println(ch);
                    k8055.clearDigitalChannel(ch);
                    subStep++;
                } else if (subStep == 16) {
                    Serial.println(F("Zapínám všechny výstupy
(setAllDigital)..."));
                    k8055.setAllDigital();
                    subStep++;
                } else if (subStep == 17) {
                    Serial.println(F("Maska 0b10101010
(writeAllDigital)..."));
                    k8055.writeAllDigital(0xAA);
                    subStep++;
                } else if (subStep == 18) {
                    Serial.println(F("Maska 0b01010101
(writeAllDigital)..."));
                    k8055.writeAllDigital(0x55);
                    subStep++;
                } else if (subStep == 19) {
                    Serial.println(F("Vypínám všechny výstupy
(clearAllDigital)..."));
                    k8055.clearAllDigital();
                    subStep = 0;
                    stateTimer = millis();
                    currentState = STATE_TEST_ANALOG_OUT;
```

```
                    Serial.println(F("\n--- 3. TEST ANALOGOVÝCH VÝSTUPŮ
(DAC1, DAC2) ---"));
                }
            }
            break;

        case STATE_TEST_ANALOG_OUT:
            if (millis() - stateTimer >= 500) {
                stateTimer = millis();

                if (subStep == 0) {
                    Serial.println(F("DAC1 = 64 (~1.25V)
[outputAnalogChannel(1, 64)]"));
                    k8055.outputAnalogChannel(1, 64);
                    subStep++;
                } else if (subStep == 1) {
                    Serial.println(F("DAC2 = 192 (~3.75V)
[outputAnalogChannel(2, 192)]"));
                    k8055.outputAnalogChannel(2, 192);
                    subStep++;
                } else if (subStep == 2) {
                    Serial.println(F("Oba DAC = 128 (~2.5V)
[outputAllAnalog(128, 128)]"));
                    k8055.outputAllAnalog(128, 128);
                    subStep++;
                } else if (subStep == 3) {
                    Serial.println(F("DAC1 MAX (255)
[setAnalogChannel(1)]"));
                    k8055.setAnalogChannel(1);
                    subStep++;
                } else if (subStep == 4) {
                    Serial.println(F("DAC2 MAX (255)
[setAnalogChannel(2)]"));
                    k8055.setAnalogChannel(2);
                    subStep++;
                } else if (subStep == 5) {
                    Serial.println(F("Oba DAC MAX (255) [setAllAnalog()]"));
                    k8055.setAllAnalog();
                    subStep++;
                } else if (subStep == 6) {
                    Serial.println(F("Vynuluji DAC1
[clearAnalogChannel(1)]"));
                    k8055.clearAnalogChannel(1);
                    subStep++;
                } else if (subStep == 7) {
                    Serial.println(F("Vynuluji DAC2
[clearAnalogChannel(2)]"));
                    k8055.clearAnalogChannel(2);
                    subStep++;
                } else if (subStep == 8) {
                    Serial.println(F("Vynuluji oba DAC
[clearAllAnalog()]"));
                    k8055.clearAllAnalog();
                    subStep = 0;
                    stateTimer = millis();
                    currentState = STATE_TEST_READBACK;
                    Serial.println(F("\n--- 4. TEST READBACK VÝSTUPŮ
(KONTROLA VNITŘNÍHO STAVU) ---"));
                }
```

```
            }
            break;

        case STATE_TEST_READBACK:
            if (millis() - stateTimer >= 400) {
                stateTimer = millis();

                if (subStep == 0) {
                    Serial.println(F("Nastavuji digitální výstupy
0x0F..."));
                    k8055.writeAllDigital(0x0F);
                    subStep++;
                } else if (subStep == 1) {
                    Serial.println(F("Nastavuji analogové výstupy DAC1=100,
DAC2=200..."));
                    k8055.outputAllAnalog(100, 200);
                    subStep++;
                } else if (subStep == 2) {
                    uint8_t digState = k8055.readBackDigitalOut();
                    uint8_t a1, a2;
                    k8055.readBackAnalogOut(a1, a2);

                    Serial.print(F("Zpětné čtení digitálních výstupů
(očekáváno 0x0F): 0x"));
                    Serial.println(digState, HEX);
                    Serial.print(F("Zpětné čtení analogových výstupů
(očekáváno DAC1=100, DAC2=200): DAC1="));
                    Serial.print(a1);
                    Serial.print(F(", DAC2="));
                    Serial.println(a2);
                    subStep++;
                } else if (subStep == 3) {
                    k8055.clearAllDigital();
                    subStep++;
                } else if (subStep == 4) {
                    k8055.clearAllAnalog();
                    subStep = 0;
                    stateTimer = millis();
                    printTimer = millis();
                    currentState = STATE_TEST_DIGITAL_IN;
                    Serial.println(F("\n--- 5. TEST DIGITÁLNÍCH VSTUPŮ (IN1
- IN5) [Trvání: 5s] ---"));
                    Serial.println(F("Stiskněte tlačítka / přiveďte signál
na vstupy IN1..IN5..."));
                }
            }
            break;

        case STATE_TEST_DIGITAL_IN:
            if (millis() - printTimer >= 500) {
                printTimer = millis();

                refreshK8055Board();

                uint8_t allDig = k8055.readAllDigital();

                Serial.print(F("Maska vstupů: 0b"));
                for (int8_t b = 4; b >= 0; b--) {
                    Serial.print((allDig & (1 << b)) ? '1' : '0');
```

```
                }

                Serial.print(F(" | Jednotlivé kanály: IN1="));
                Serial.print((allDig & 0x01) ? 1 : 0);
                Serial.print(F(" IN2="));
                Serial.print((allDig & 0x02) ? 1 : 0);
                Serial.print(F(" IN3="));
                Serial.print((allDig & 0x04) ? 1 : 0);
                Serial.print(F(" IN4="));
                Serial.print((allDig & 0x08) ? 1 : 0);
                Serial.print(F(" IN5="));
                Serial.println((allDig & 0x10) ? 1 : 0);
            }

            if (millis() - stateTimer >= 5000) {
                stateTimer = millis();
                printTimer = millis();
                currentState = STATE_TEST_ANALOG_IN;
                Serial.println(F("\n--- 6. TEST ANALOGOVÝCH VSTUPŮ (AD1,
AD2) [Trvání: 5s] ---"));
                Serial.println(F("Otáčejte potenciometry AD1 a AD2 na desce
K8055..."));
            }
            break;

        case STATE_TEST_ANALOG_IN:
            if (millis() - printTimer >= 500) {
                printTimer = millis();

                refreshK8055Board();

                uint8_t ad1, ad2;
                k8055.readAllAnalog(ad1, ad2);

                Serial.print(F("Analogový vstup AD1: "));
                Serial.print(ad1);
                Serial.print(F(" | AD2: "));
                Serial.println(ad2);
            }

            if (millis() - stateTimer >= 5000) {
                subStep = 0;
                stateTimer = millis();
                printTimer = millis();
                currentState = STATE_TEST_COUNTERS;
                Serial.println(F("\n--- 7. TEST ČÍTAČŮ (COUNTER 1 & 2)
[Trvání: 5s] ---"));
            }
            break;

        case STATE_TEST_COUNTERS:
            if (subStep == 0) {
                Serial.println(F("Nastavuji filtr zákmitů (debounce 10ms)
pro Čítač 1..."));
                k8055.setCounterDebounceTime(1, 10);
                subStep++;
                stateTimer = millis();
            } else if (subStep == 1 && millis() - stateTimer >= 250) {
```

```
                Serial.println(F("Nastavuji filtr zákmitů (debounce 10ms)
pro Čítač 2..."));
                k8055.setCounterDebounceTime(2, 10);
                subStep++;
                stateTimer = millis();
            } else if (subStep == 2 && millis() - stateTimer >= 250) {
                Serial.println(F("Vynuluji Čítač 1..."));
                k8055.resetCounter(1);
                subStep++;
                stateTimer = millis();
            } else if (subStep == 3 && millis() - stateTimer >= 250) {
                Serial.println(F("Vynuluji Čítač 2..."));
                k8055.resetCounter(2);
                Serial.println(F("Čítače vynulovány. Přiveďte impulzy na
vstupy IN1 a IN2..."));
                subStep++;
                stateTimer = millis();
                printTimer = millis();
            } else if (subStep == 4) {
                if (millis() - printTimer >= 500) {
                    printTimer = millis();

                    refreshK8055Board();

                    uint32_t cnt1 = k8055.readCounter(1);

                    Serial.print(F("Hodnota Čítače 1 (IN1): "));
                    Serial.print(cnt1);
                    Serial.print(F(" | Hodnota Čítače 2 (IN2): "));
                    Serial.println(k8055.readCounter(2));
                }

                if (millis() - stateTimer >= 5000) {
                    subStep = 0;
                    stateTimer = millis();
                    printTimer = millis();
                    currentState = STATE_TEST_READ_ALL;
                    Serial.println(F("\n--- 8. TEST HROMADNÉHO ČTENÍ
(readAll) [Trvání: 5s] ---"));
                }
            }
            break;

        case STATE_TEST_READ_ALL:
            if (millis() - printTimer >= 500) {
                printTimer = millis();

                uint8_t digIn, an1, an2;
                uint32_t cnt1, cnt2;

                k8055.readAll(digIn, an1, an2, cnt1, cnt2);

                Serial.print(F("[readAll] Vstupy (bin): 0b"));
                for (int8_t b = 4; b >= 0; b--) {
                    Serial.print((digIn & (1 << b)) ? '1' : '0');
                }
                Serial.print(F(" | AD1: "));
                Serial.print(an1);
                Serial.print(F(" | AD2: "));
```

```
                Serial.print(an2);
                Serial.print(F(" | Č1: "));
                Serial.print(cnt1);
                Serial.print(F(" | Č2: "));
                Serial.println(cnt2);
            }

            if (millis() - stateTimer >= 5000) {
                stateTimer = millis();
                currentState = STATE_TEST_COMPLETE;
            }
            break;

        case STATE_TEST_COMPLETE:

Serial.println(F("\n============================================="));
            Serial.println(F("  [DOKONČENO] Všechny funkce úspěšně
otestovány!"));

Serial.println(F("============================================="));

            currentState = STATE_TEST_END;

            break;

        // ------------------------------------------------------------------
----
        // 11. STOP
        // ------------------------------------------------------------------
----
        case STATE_TEST_END:
          break;

    }

    wdt_reset();
}
```