

Entprellte Eingabe (Debouncing)

Die BiDiBOne-Basis-Firmware enthält eine einfache Funktionalität für entprellte Eingaben wie Schalter und Taster. Diese Eingabeinformationen können an den BiDiBus gesendet und für die Steuerung von Makros verwendet werden.

Hintergrund

Mechanische Schaltelemente wie Taster, Schalter oder auch Relais prellen beim Umschalten, d.h. sie schalten nicht plötzlich von einem in den anderen Zustand um, sondern wechseln im Millisekundenbereich ihren Zustand bis zur gewollten Ruhelage. Dieses Verhalten wird als „Prelen“ bezeichnet.

Da einige Millisekunden für den Prozessor eine Ewigkeit sind, würde eine nicht optimierte Software jede Änderung dieser Schaltzustände auswerten und zu Fehlschlüssen gelangen. Probates Mittel dagegen ist das so genannte Entprellen (engl. debounce). In analogen Systemen wird das Prelen u.A. mittels Kondensatoren unterbunden. In unserer BiDiBOne-Basis entprellen wir diese Signale Software technisch.

Resourcenbelegung

Die „entprellte Eingabe“ bildet die Eingangsinformationen auf die [Keyboard-Ereignisse im Event-System](#) ab.

Je nach Anzahl der Eingaben (ein bis acht) werden die Ereignisse `IN_DOWN0` bzw. `IN_UP0` bis `IN_DOWN7` bzw. `IN_UP7` verwendet. Die dürfen dann nicht mehr anderweitig verwendet werden.

Die Eingänge werden als `INPUT` beginnend bei 0 aufsteigend im System geführt.

Konfiguration

Die in der BiDiBOne-Basis-Firmware implementierte „Entprellte Eingabe“ wird durch Konfiguration in `addon_model.h` aktiviert.

```
//=====
// Switchboard defines (Simple keyboard with debouncing and eventing to macro engine)
// These are the first INPUTS!
// You can collect 8 input bits in sum from 2 ports but have to shift them into 1 "byte".
// - You can shift to higher (positive) or lower (negative value) position.
// - We do not check shifted position - be careful!
// - If you overlap bits we interpret them as one key
// - Least bit becomes D0 and so on means INPUT0...7(max)
// - D0 will become KEY0 and D7 will become KEY7
// - EXTERNAL2 is activated if EXTERNAL1 is enabled only!
// Example:
// #define EXTERNAL1_INPUT_ENABLED      TRUE    // FALSE|TRUE: no switches| switches with following conditions
// #define EXTERNAL1_INPUT_PORT        PORTF    // PORTA...PORTn: This is the port where we read bits
// #define EXTERNAL1_INPUT_MASK        0x0F    // 4 least significant bits
// #define EXTERNAL1_INPUT_SHIFT        0       // not shifted
//
// #define EXTERNAL2_INPUT_ENABLED      TRUE    // FALSE|TRUE: no switches| switches with following conditions
// #define EXTERNAL2_INPUT_PORT        PORTA    // PORTA...PORTn: This is the port where we read bits
// #define EXTERNAL2_INPUT_MASK        0x0F    // 4 least significant bits
// #define EXTERNAL2_INPUT_SHIFT        4       // shift them to high nibble
//=====
#define EXTERNAL_INPUT_ACTIVE_TO_GND  TRUE    // TRUE|FALSE: (all) pins to GND

#define EXTERNAL1_INPUT_ENABLED      TRUE    // FALSE|TRUE: no switches| switches with following conditions
#define EXTERNAL1_INPUT_PORT        PORTF    // PORTA...PORTn: This is the port where we read bits
#define EXTERNAL1_INPUT_MASK        0x0F    // 4 least significant bits
#define EXTERNAL1_INPUT_SHIFT        0       // not shifted

#define EXTERNAL2_INPUT_ENABLED      FALSE    // FALSE|TRUE: no switches| switches with following conditions
#define EXTERNAL2_INPUT_PORT        PORTC    // PORTA...PORTn: This is the port where we read bits
#define EXTERNAL2_INPUT_MASK        0xF0    // 4 least significant bits
#define EXTERNAL2_INPUT_SHIFT        4       // shift them to high nibble
//=====
```

1. Es können maximal 8 Eingänge von insgesamt 2 Ports konfiguriert werden
2. Die Pins eines Ports sollten zusammenhängend sein
3. Die beiden Gruppen müssen ohne Überlappung in eine 8-Bit-Variable verschoben werden
4. Das niederwertigste Bit wird als INPUT0 interpretiert
5. EXTERNAL2 wird nur mit EXTERNAL1 aktiviert

Definition	Wert	Bedeutung
EXTERNAL_INPUT_ACTIVE_TO_GND	TRUE/FALSE	Aktivlage aller Eingänge
EXTERNAL1_INPUT_ENABLED	FALSE/TRUE	deaktiviert/aktiviert die Gruppe
EXTERNAL1_INPUT_PORT	PORTA...PORTF	Port von dem gelesen wird
EXTERNAL1_INPUT_MASK	0x00...0xFF	Maske für die relevanten Bits
EXTERNAL1_INPUT_SHIFT	0	ohne Verschiebung
	>0	Verschiebung nach links (positiv)
	<0	Verschiebung nach rechts (negativ)

Analog dazu wird **EXTERNAL2...** definiert.

Um Konflikte zu vermeiden (siehe [AddOn-Software einbinden in BiDiBOne](#)), sollten auch hier die angeforderten Portpins unbedingt gekennzeichnet werden:

```

#define EXTERNAL_INPUT_ACTIVE_TO_GND    TRUE    // TRUE|FALSE: (all)
#define EXTERNAL1_INPUT_ENABLED          TRUE    // FALSE|TRUE: no :
#define EXTERNAL1_INPUT_PORT             PORTF   // PORTA...PORTn: 1
#define EXTERNAL1_INPUT_MASK             0x0F    // 4 least signific
#define EXTERNAL1_INPUT_SHIFT            0        // not shifted

#define EXTERNAL2_INPUT_ENABLED          // FALSE|TRUE: no :
#define EXTERNAL2_INPUT_PORT             // PORTA...PORTn: 1
#define // 4 least signific
#define them to hi

//////// Check resources for EXTERNALs //////////
#ifndef I_NEED_PORTF0
#define I_NEED_PORTF0
#else
#error Port conflict F0
#endif
#ifndef I_NEED_PORTF1
#define I_NEED_PORTF1
#else
#error Port conflict F1
#endif
#ifndef I_NEED_PORTF2
#define I_NEED_PORTF2
#else
#error Port conflict F2
#endif
#ifndef I_NEED_PORTF3
#define I_NEED_PORTF3
#else
#error Port conflict F3
#endif
//////// Check resources for EXTERNALs //////////

```

Softwareeinbindung

Neben der Konfiguration ist die Weiterleitung einer dieser Eingangs-Informationen mit der Funktion: `check_debounced_input` aus dem Modul `keyboard.c` möglich.

```

/**
 * \brief Checks the input with the given number and returns result.
 *        This function is called by the macro engine to start a macro.
 *
 * \param number    number of input (0...7)
 *
 * \return uint8_t  result of check
 * \retval          TRUE  input is active
 * \retval          FALSE input is not active
 */
uint8_t check_input_addon(uint8_t number)
{
    return check_debounced_input(number);
}

```

Das kann z.B. in der Callback-Funktion `check_input_addon` aus dem Modul `addon.c` passieren.

```
/**
 * \brief Checks the input with the given number and returns result.
 *        This function is called by the macro engine to start a macro.
 *
 * \param number    number of input (0...7)
 *
 * \return uint8_t  result of check
 * \retval          TRUE   input is active
 * \retval          FALSE  input is not active
 */
uint8_t check_input_addon(uint8_t number)
{
    return check_debounced_input(number);
}
```

Zusammenfassung

1. Konfiguration in `addon_model.h`
2. Abfrage und Weiterleitung in `check_input_addon` mit `check_debounced_input`

From:
<https://forum.opendcc.de/wiki/> - BiDiB Wiki

Permanent link:
https://forum.opendcc.de/wiki/doku.php?id=softwarebausteine:entprellte_eingabe

Last update: **2014/09/01 12:34**

