

Protokoll 4

Praktikum Technische Informatik

Tas Soti, Richard Wilhelm, Naja v. Schmude

21. Juni 2008

1 Per Anruf digitale I/O Portleitungen schalten

1.1 Vorbereitung

Als erstes wird sich über CLIR und CLIP informiert, und wie man diese Funktionen mittels AT-Kommando schaltet. CLIR steht hierbei für "*Calling Line Identification Restriction*" und ist für die Unterdrückung der Rufnummerübermittlung für abgehende Anrufe zuständig. Der passende AT-Befehl lautet AT+CLIR und erwartet einen Parameter: Mit 1 wird die eigene Rufnummer mit übertragen, mit 2 wird sie unterdrückt.

Quasi als Pendant gibt es CLIP, "*Calling Line Identification Presentation*", was entscheidet, ob bei eingehenden Anrufen die Anrufernummer übertragen werden soll und auf dem Modul angezeigt oder nicht. Mittels AT+CLIP wird dann entsprechend des übergebenen Parameters CLIP aktiviert (1) bzw. deaktiviert (0). Bei aktiviertem CLIP erscheint auf dem Modul bei eingehendem Anruf dann zusätzlich zu dem RING noch eine Meldung in der Art +CLIP: "0146290800",129,1,, "FRED", wobei der erste Parameter hier die Telefonnummer angibt.

Da wir ja dann in der Aufgabe die I/O-Portleitungen schalten wollen, muss man sich entsprechend noch dazu die Informationen beschaffen, wie man diese denn steuert.

Wie üblich, muss man sich auch hier erstmal für eine oder mehrere Portleitungen mittels `adl_ioSubscribe` registrieren, um auf diesen später dann lesen bzw. schreiben zu können. Der Befehl erwartet fünf Parameter, der erste gibt eine Bitmaske an, die die zu registrierenden Portleitungen angibt. Wir wollen ja auf GPO0 verwenden, daher brauchen wir nachher die Maske `ADL_IO_Q25X1_GPO_0` entsprechend unseres Moduls. Der zweite Parameter gibt an, in welche Richtung (also Input oder Output) die Leitung registriert werden soll. Für uns ist das allerdings nicht wichtig, da wir sowieso nur eine Outputleitung nutzen. Der dritte Parameter gibt an, was der Standardwert sein soll für den Output, also 0 oder 1. Der nächste Parameter definiert die Polling-Zeit, das ist für uns aber irrelevant, weil wir kein Input haben, daher kommt hier eine 0 hin. Und als letztes brauchen wir natürlich wieder einen Handler, der nach jedem Polling aufgerufen wird. Da wir kein Polling haben, ist hier `NULL`.

Wenn wir jetzt die Portleitung registriert haben, können wir mit `adl_ioWrite` auf ihr schreiben. Dazu wird nur die registrierte Leitung benötigt, wieder die Bitmaske um die richtige anzusteuern und natürlich das, was geschrieben werden soll.

1.2 Aufgaben und Durchführung

- Das Modul lässt sich die Rufnummern eingehender Anrufe übermitteln und übermittelt die eigene Rufnummer bei ausgehenden Anrufen.

```
void a10_apply(void) {
    adl_atCmdCreate("AT+CLIP=1", FALSE, (adl_atRspHandler_t)
        NULL, NULL); // CLIP einschalten
    adl_atCmdCreate("AT+CLIR=1", FALSE, (adl_atRspHandler_t)
        NULL, NULL); // CLIR einschalten
    adl_atUnSoSubscribe("+CLIP", (adl_atUnSoHandler_t)
        a10_telenummerAuslesen_Handler);
}

/* Wird eine Telefonnummer übertragen, wird sie ausgelesen
*/
bool a10_telenummerAuslesen_Handler(adl_atUnsolicited_t *
    parameter) {
    wm_strGetParameterString(a10param, parameter->
        StrData, 1); // Nummer auslesen
    adl_atSendResponse(ADL_AT_RSP, a10param); // Telenr
        . ausgeben
    return FALSE;
}
```

Zunächst wird über `adl_atCmdCreate` der `AT+CLIP=1` und `AT+CLIR=1` Befehl abgesetzt, damit CLIR und CLIP aktiviert werden. Damit wir nun auch die eingehende Telefonnummer übertragen bekommen, müssen wir mit `adl_atUnSoSubscribe` uns für das Event `+CLIP` registrieren. Wenn jetzt ein Anruf kommt, dann wird der Handler `a10_telenummerAuslesen_Handler` aufgerufen.

Hier wird jetzt mit der altbekannten Methode der erste Parameter ausgelesen (das ist ja gerade die Telefonnummer) und in der globalen Variablen `a10param` abgespeichert.

- Nur eine fest eingespeicherte Rufnummer schaltet die LED. Alle anderen Anrufe haben keinen Einfluss.

```
void a10_apply(void) {
    a10status &= (~ADL_IO_Q25X1_GPO_0); // Lämpchen initial
        ausschalten
    a10io = adl_ioSubscribe(ADL_IO_Q25X1_GPO_0, 0, 0, 0, (
        adl_ioHdlr_f) NULL); // Lampe GPO_0 anmelden
    ...
}
```

```

/* Wird eine Telefonnummer übertragen, wird sie ausgelesen
, verglichen und bei Übereinstimmung das Lämpchen
umgeschaltet */
bool a10_telenummerAuslesen_Handler(adl_atUnsolicited_t *
parameter) {
    wm_strGetParameterString(a10param, parameter->
        StrData, 1); // Nummer auslesen
    adl_atSendResponse(ADL_AT_RSP,a10param); // Telenr
        . ausgeben
    if(wm_strcmp(a10param,a10telenummer) == 0) { //
        Telefonnummern sind gleich
        adl_atSendResponse(ADL_AT_RSP,"Gleiche_
            Nummern");
        if(a10status & ADL_IO_Q25X1_GPO_0) //
            Lampe ist an -> ausschalten
            a10status &= ~ADL_IO_Q25X1_GPO_0;
        else // Lampe ist aus -> anschalten
            a10status |= ADL_IO_Q25X1_GPO_0;
        adl_ioWrite(a10io,ADL_IO_Q25X1_GPO_0 ,
            a10status); // neue Status schreiben
    }
    return FALSE;
}

```

Zunächst müssen wir erstmal das Lämpchen mit `adl_ioSubscribe` anmelden. Nun wird die Telefonnummer-Auslesen-Methode entsprechend erweitert. Jetzt wird bei jedem Anruf geprüft, ob die anrufende Nummer mit unser fest eingespeicherten Nummer (in `a10telenummer`) übereinstimmt. Ist dies der Fall, wird geguckt, ob die Lampe gerade an oder aus ist. Da es sich um eine Leitung handelt, kann man nicht einfach nur 0 oder 1 schicken, sondern muss geschickt mit der Bitmaske hantieren, um die entsprechenden Bits zu kippen. Wenn die Lampe z.B. an ist, dann erkennt man das daran, dass wir ne logische Eins bekommen. Daher können wir mit dem logischen Und (&) überprüfen, ob der Status gerade an ist, in dem wir mit der Bitmaske verknüpfen. Um die Bits zu kippen zum Ausschalten, muss man dann mit der negierten Bitmaske verunden. Beim Anschalten müssen wir dann einfach nur mit der Bitmaske verodern. Wenn wir jetzt den Status geändert haben, muss er natürlich wieder per `adl_ioWrite` zurückgeschrieben werden.

- Der Anruf mit der passenden Nummer wird abgewiesen. Danach wird die LED umgeschaltet (d.h wenn sie nicht leuchtet eingeschaltet und umgekehrt). Anschließend wird der Anrufer zur Bestätigung nach 10 Sekunden zurückgerufen.

```

/* Wird eine Telefonnummer übertragen, wird sie ausgelesen
, verglichen und bei Übereinstimmung das Lämpchen
umgeschaltet und die Nummer angerufen */

```

```

bool a10_telenummerAuslesen_Handler(adl_atUnsolicited_t *
    parameter) {
    wm_strGetParameterString(a10param, parameter->
        StrData, 1); // Nummer auslesen
    adl_atSendResponse(ADL_AT_RSP,a10param); // Telenr
        . ausgeben
    if(wm_strcmp(a10param,a10telenummer) == 0) { //
        Telefonnummern sind gleich
        adl_atSendResponse(ADL_AT_RSP,"Gleiche_
            Nummern");
        adl_callHangup(); // legt auf
        if(a10status & ADL_IO_Q25X1_GPO_0) //
            Lampe ist an -> ausschalten
            a10status &= ~ADL_IO_Q25X1_GPO_0;
        else // Lampe ist aus -> anschalten
            a10status |= ADL_IO_Q25X1_GPO_0;
        adl_ioWrite(a10io,ADL_IO_Q25X1_GPO_0 ,
            a10status); // neue Status schreiben
        // timer starten und rückruf nach 10
            Sekunden
        a10timer = (adl_tmr_t *) adl_tmrSubscribe(
            FALSE, 100, ADL_TMR_TYPE_100MS, (
                adl_tmrHandler_t) a10Timer_Handler);
    }
    return FALSE;
}

void a10Timer_Handler(u8 Id) {
    adl_tmrUnSubscribe(a10timer, a10Timer_Handler ,
        ADL_TMR_TYPE_100MS);
    adl_callSetup(a10telenummer,ADL_CALL_MODE_VOICE);
}

```

Per `adl_callHangup()` wird einfach aufgelegt. Dann ändern wir wie oben beschrieben den Status. Hinzu kommt jetzt nur noch ein 10-Sekunden-Timer, der dann per `adl_calSetup` die fest eingespeicherte Telefonnummer per Voice-Call anruft.