Technische Informatik IV: Praktikum

Protokoll zu Aufgabe 11

von Alexander Steen, Max Wisniewski

Vorbereitung

Zur Vorbereitung haben wir uns mit dem Abschnitt *Flash* des *ADL User Guide*, sowie dem Rahmenwerk *file.c* auseinandergesetzt.

Aufgaben

1. AT-Kommandos `AT+OWNER`, `AT+SIMFILE`, `AT+SIMCHANGE` erstellen
2. Überprüfung der gespeicherten Nummer (Vergleich mit der tatsächlichen Nummer) implementieren
3. SMS mit den relevanten Daten verschicken, falls der Test (2) fehlschlägt

Dokumentation

Einige Funktionen aus dem Rahmenwerk `file.c`:

file_exists Diese funktion überprüft, ob eine Datei bereits existiert und gibt im Falle ihrer Existenz die Größe der Datei in Byte zurück, 0 sonst. Als Parameter bekommt die Funktion den Namen der Datei.

file_erase Löscht den Inhalt einer Datei, falls diese existiert. Als Parameter erhält die Funktion den Namen der Datei, die gelöscht werden soll.

file_write Ein Aufruf von `file_write(ascii * name, ascii * data, u16 length)` schreibt den Inhalt `data` in die Datei mit dem Namen `name`. Dabei muss `length` auf die Länge des zu schreibenden Inhalts gesetzt werden.

file_read Gibt den einen `ascii`-Pointer auf den Inhalt der Datei zurück, falls sie existiert. Sonst `NULL`. Auf diese Weise gelesene Inhalte müssen mit `adl_memRelease` explizit wieder freigegeben werden.

Durchführung und Auswertung

Zunächst legen den Handler des Kommandos `AT+OWNER` `ownerHandler` an. Dieser soll beim Ausführen eine Datei "owner.txt" anlegen, die als Inhalt die als Parameter übergebene Telefonnummer enthält. Existiert die Datei bereits, wird sie gelöscht und neu angelegt.

```
void ownerHandler(adl_atCmdPreParser_t *param) {
    if (param->Type == ADL_CMD_TYPE_PARA) {
        if (file_exist("owner.txt")) {
            // Existiert schon, also neu anlegen
            file_erase("owner.txt");
        }
        ascii owner[20];
        wm_strGetParameterString(owner, param->StrData, 1);
        file_write("owner.txt", owner, 20); // Inhalt schreiben
        // Kontrollausgaben
        adl_atSendResponse(ADL_AT_RSP, "\r\n␣owner.txt␣mit␣Nummer␣");
        adl_atSendResponse(ADL_AT_RSP, owner);
        adl_atSendResponse(ADL_AT_RSP, "␣angelegt␣\r\n");
    }
}
```

Auch für das designierte AT-Kommando `AT+SIMFILE` legen wir einen Handler namens `simHandler` an. Dieser versucht das Telefonbuch auf "ON" zu stellen. Da das nicht immer glückt, überprüfen wir mit einer AT-Abfrage, was die aktuelle Telefonbuchselektion ist. Der Handler `phonebookSwitchHandler` kümmert sich dann um die Antwort dieser Abfrage.

```
void simHandler(adl_atCmdPreParser_t *param) {
    if (param->Type == ADL_CMD_TYPE_ACT) {
        adl_atCmdSend("AT+CPBS=\"ON\"", NULL, NULL);
        adl_atCmdSend("AT+CPBS?", phonebookSwitchHandler,
                                        "+CPBS:", NULL);
    }
}
```

Falls der Wechsel zu ON glückt, wird die SIM-Nummer angefordert (Eintrag 1 im Telefonbuch ON) und der `writeSIMNumberHandler` mit dem Schreiben der Nummer beauftragt, der die Antwort der Telefonbuch-Abfrage enthält.

```
bool phonebookSwitchHandler(adl_atResponse_t * param)   {
    ascii phonebook[5];
    wm_strGetParameterString(phonebook, param->StrData, 1);
    if (wm_strcmp("ON",phonebook) == 0) {
        // Wechsel zu ON glückte: Weitermachen
        if (file_exist("sim.txt")) {
            // Datei existiert schon: Neu anlegen
            file_erase("sim.txt");
        }
        // Nummer vom Telefonbuch anfordern (SIM-Nummer)
        adl_atCmdSend("AT+CPBR=1", writeSIMNumberHandler, "+CPBR:", NULL);
    } else {
        // Wechsel klappte nicht, Fehlermeldung ausgeben
        adl_atSendResponse(ADL_AT_RSP,
        "\r\n +SIMFILE ERROR: Wechsel zum Telefonbuch ON nicht erfolgreich!\r\n");
        return false;
    }
}
```

In dem `writeSIMNumberHandler` wird dann nur noch die Datei geschrieben, das Telefonbuch wieder zurück auf SM gesetzt und eine Kontrollausgabe getätigt:

```
bool writeSIMNumberHandler(adl_atResponse_t * param)    {
    ascii number[20];
    // Nummer aus Parameter holen und schreiben
    wm_strGetParameterString(number, param->StrData, 2);
    file_write("sim.txt", number, 20);

    // Telefonbuch zurück auf SM setzen
    adl_atCmdSend("AT+CPBS=\"SM\"", NULL, NULL);

    // Kontroll-Ausgabe
    adl_atSendResponse(ADL_AT_RSP, "\r\n In sim.txt geschrieben: ");
    adl_atSendResponse(ADL_AT_RSP, number);
    adl_atSendResponse(ADL_AT_RSP, "\r\n");
    return false;
}
```

Wiederum analog zu `ownerHandler` legen wir für das neue Kommando `AT+SIMCHANGE` einen Handler namens `simChangeHandler` an. Dieser enthält Parameter, nämlich die neue Nummer und kann mit `AT+SIMCHANGE=<NeueNummer>` aufgerufen werden.

```
void simChangeHandler(adl_atCmdPreParser_t *param) {
    if (param->Type == ADL_CMD_TYPE_PARA) {
        if (file_exist("sim.txt")) {
        // Datei existiert schon, neu anlegen
            file_erase("sim.txt");
        }
        ascii newsim[20];
        wm_strGetParameterString(newsim, param->StrData, 1);

        file_write("sim.txt", newsim, 20); // Inhalt schreiben
        // Kontrollausgaben
        adl_atSendResponse(ADL_AT_RSP, "\r\n sim.txt auf Nummer ");
        adl_atSendResponse(ADL_AT_RSP, newsim);
        adl_atSendResponse(ADL_AT_RSP, " verändert \r\n");
    }
}
```

Um die AT-Funktionen nun bereitzustellen, registrieren wir sie am Anfang der `main`-Funktion mit ihren entsprechenden Eigenschaften. Falls nun die Textdateien vorhanden sind, leiten wir die Überprüfung ein, in dem wir das Telefonbuch auf ON wechseln (um die SIM-Nummer herauszufinden). Da dies wie oben fehlschlagen kann, brauchen wir wiederum einen Handler, der überprüft, ob der Wechsel erfolgreich war. [1]

```
s8 smsHandle;
ascii momentOwner[20];
...
void main_task(void) {
    // Neue AT-Kommandos registrieren
    adl_atCmdSubscribe("AT+OWNER", ownerHandler, ADL_CMD_TYPE_PARA | 0x0011);
    adl_atCmdSubscribe("AT+SIMFILE", simHandler, ADL_CMD_TYPE_ACT);
    adl_atCmdSubscribe("AT+SIMCHANGE", simChangeHandler, ADL_CMD_TYPE_PARA | 0x0011);
    // Existieren die Dateien? Wenn ja: Überprüfung starten
    if (file_exist("sim.txt") && file_exist("owner.txt")) {
        // SMS Handler anlegen, falls wir eine SMS schicken müssen
        smsHandle = adl_smsSubscribe(leersmshandler, leersmsCtrlHandler, ADL_SMS_MODE_TEXT);
        // Auf ON wechseln
        adl_atCmdSend("AT+CPBS=\"ON\"", NULL, NULL);
        adl_atCmdSend("AT+CPBS?", anotherPhonebookSwitchHandler,
                                        "+CPBS:", NULL);
    }
}
```

Wenn der Wechsel zum Telefonbuch ON nicht erfolgreich war, können wir die Überprüfungsfunktionalität nicht anbieten, also tun wir nichts. Sollte es aber erfolgreich gewesen sein, dann beauftragen wir `checkhandler` die Gleichheit der Nummern zu überprüfen.

```
bool anotherPhonebookSwitchHandler(adl_atResponse_t * param)    {
    ascii phonebook[5];
    wm_strGetParameterString(phonebook, param->StrData, 1);
    if (wm_strcmp("ON",phonebook) == 0) {
        // Wechsel zu ON glückte: Nummer vom Telefonbuch anfordern und überprüfen
        adl_atCmdSend("AT+CPBR=1", checkhandler, "+CPBR:", NULL);
    } else { // Wechsel klappte nicht, also Funktionalität ignorieren
        return false;
    }
}
```

[1] Das hätte man wahrscheinlich viel eleganter mit Funktionspointer lösen können, da die beiden Wechsel-Funktionen eigentlich analog arbeiten. Aber da wir ziemlich schlecht in C sind, haben wir uns für die ekelige,aber funktionierende Version entschieden - Sorry ;-)

Nun muss der `checkhandler` die Gleichheit überprüfen und bei Ungleichheit eine SMS schicken. Dafür überprüfen wir zunächst die Nummern und fordern im Falle der Gleichheit durch `AT+CCED=0,1` die Daten (MCC, MNC, ...) an. Diese Daten verarbeitet dann eine weitere Callbackfunktion namens `posHandler`, die auch den SMS-Text zusammenbaut und die SMS endgültig schickt. Die Variable `momentOwner` ist global und erlaubt dem `posHandler` auf die Nummer zuzugreifen.

```
bool checkhandler(adl_atResponse_t * param) {
    wm_strGetParameterString(momentOwner, param->StrData, 2);
    adl_atSendResponse(ADL_AT_RSP, momentOwner);
    // Aktuelle Nummer auslesen
    ascii * sim = file_read("sim.txt");
    if (wm_strcmp(sim, momentOwner) != 0) {
        // Nummern stimmen nicht überein: Diebe!
        adl_atCmdSend("AT+CCED=0,1", posHandler, "+CCED:", NULL);
    }
    return false;
}

bool posHandler(adl_atResponse_t * param) {
    ascii sendBuffer[40];
    ascii mccBuffer[3]; ascii mncBuffer[2];
    ascii lacBuffer[4]; ascii ciBuffer[4];
    // Parameter holen und speichern (Antwort von +CCED)
    wm_strGetParameterString(mccBuffer, param->StrData, 1);
    wm_strGetParameterString(mncBuffer, param->StrData, 2);
    wm_strGetParameterString(lacBuffer, param->StrData, 3);
    wm_strGetParameterString(ciBuffer, param->StrData, 4);
    // Owner-Nummer lesen
    ascii * realowner = file_read("owner.txt");
    // Nachricht zusammenbauen
    wm_sprintf(sendBuffer, "Handy␣weg.␣Neue␣Nummer:␣%s,␣Koord:␣%s␣%s␣%s␣%s",
        momentOwner, mccBuffer, mncBuffer, lacBuffer, ciBuffer);
    // SMS schicken!
    adl_smsSend(smsHandle, realowner, sendBuffer, ADL_SMS_MODE_TEXT);
    // Kontrollausgabe
    adl_atSendResponse(ADL_AT_RSP, sendBuffer);
    return false;
}
```