Praxis mit den ESB-Knoten

Der watchdog:

startWatchdog() - setzt den Watchdog-Timer zurück.

Wurde dies etwa 1 Sekunde lang nicht gemacht, so nimmt der Watchdog an, dass der Knoten bzw. das Programm abgestürzt ist und führt einen Rest aus.

Input/Output:

sendPacket(char to, char type, char num, int length, char* data) – Funkt Nachricht

to: Die ID des Empfängers oder BROADCAST
type: SENSOR_PACKET (andere machen für uns keinen Sinn)
num: Nummer des Packetes
length: Länge der Nachricht
data: Zeiger auf den Anfang des Daten-Arrays

Adresse im Hex-Format: 0x0029

Bit 1: gesetzt = LED-Red aus / gelöscht = LED-Red an
Bit 2: gesetzt = LED-Green aus / gelöscht = LED-Green an
Bit 3: gesetzt = LED-Yellow aus / gelöscht = LED-Yellow an
Bit 4: gesetzt = Beeper nervt / gelöscht = Beeper aus

Bemerkung: Ein gesetztes Bit 4 entspricht einer numerischen 8

sendRS232String(char* text) – gibt den in text gespeicherten String auf dem Terminal aus
sendRS232Char(char* data, int length) – gibt eine Anzahl 'length' Bytes aus dem Puffer im Terminal aus
sendRS232Int(int value) – gibt value auf dem Terminal aus

Callback Funktion für Nachrichten vom Terminal:

registerCallback(C_SERIAL, handleSerial)

... bedeuete, dass die eigene Funktion void handleSerial() aufgerufen wird, wenn über das Terminal ein vollständiger String eingegeben wurde.

Der String steht dann unter der globalen Variable serial_line[] Null-terminiert zur Verfügung.

Weitere Variablen (wird verm. nicht benötigt):

serial_pos = aktuelle Anzahl Zeichen im Puffer
serial_flags = Zustand der ser. Schnittstelle z. B. SERIAL_LINE_RECEIVED

Praxis mit den ESB-Knoten

Organisation des ankommenden Pakets:

Byte 1: (to) – ID des Adressaten (evtl. 255 für Broadcast)
Byte 2: (from) – ID des Senders
Byte 3: (type) – Typ des Pakets (hier meist SENSOR_PACKET)
Byte 4: (num) – (Fortlaufende) Nummer des Pakets / so sind Verluste leicht erkannbar
Byte 5: (upper length) – die oberen 8 Bit der 16-Bit Länge (Most Significant Byte)
Byte 6: (lower length) – die unteren 8 Bit der 16-Bit Länge (Least Significant Byte)
Byte 7...: (data...) - eine Anzahl length an Datenbytes

Bemerkung: Byte 1 wird mit data[0] adressiert, wenn der Zeiger data auf den Anfang des Pakets zeigt.

Wichtige globale Variablen:

int fiveMsTimer : Zeitgeber, der sich von selbst inkrementiert
int cdCount : Carrier-Sense (0 = freier Kanal, sonst belegt)

Sensordaten : sensorData.micCount / sensorData.micValue / sensorData.vibCount / sensorData.lightCount
sensorData.id / sensorData.temperature / transmitPower / rxReceiveLimit

Vorbereitungen:

Wechseln der Konsolen mit Alt+F1 bis F4

Konsole 4: msp430-gdbproxy msp430
Konsole 3: minicom
Konsole 2: cd tmp/msp430/userapp
nano src/userapp.c (für Fortgeschrittene ist auch der emacs verfügbar)
Konsole 1: cd tmp/msp430/userapp
make um das Programm zu übersetzen
make flash um das Programm auf die Knoten zu übertragen
1. flash geht meist nicht, dann raus mit q (RETURN) und nochmal make flash (Cursor up)

Text erzeugen:

char text[80]; // erzeugt Speicher für 80 Zeichen
sprintf(text, „Gibt eine Zahl aus %d \n\r“, value);