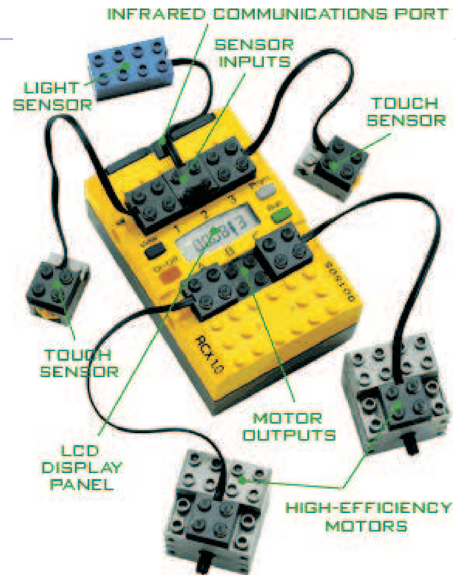
K 11. Übung

K.1 Überblick

- Hardware
  - RCX
  - Hitachi H8 / 3292
- LegOS
  - Überblick
  - Environment
  - API
    - **`http://legos.sourceforge.net/docs/CommandRef.html`**
- Aufgabe 7

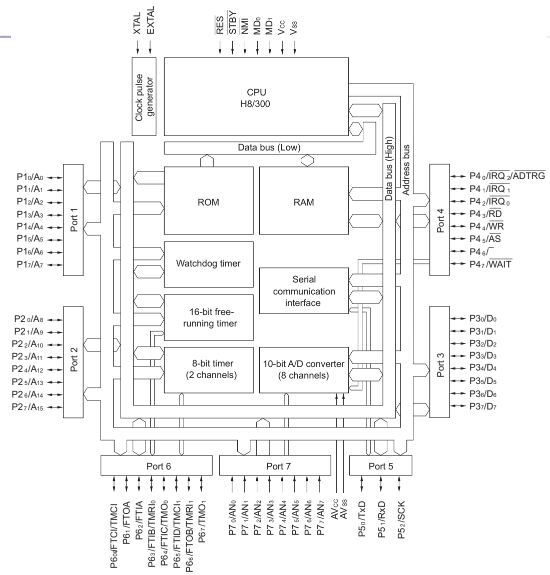
K.2 RCX

- liegt dem *Lego Robotics Invention System* bei.
- Interaktion mit der Umwelt:
  - 3 Sensoren
  - 3 Aktoren
  - 4 Knöpfe
  - 4-5 stelligs Display
  - IR-Schnittstelle
- enthält einen Hitachi H8/3292 Microcontroller



K.3 H8 / 3292

- Familie H8/3297
- 16 bit Prozessor (H8/300)
- 16 KByte ROM
- 512 Byte int. RAM
- im RCX zusätzl. 32 KByte ext. RAM



K.4 LegOS

- Firmware für Lego RCX
  - dynamisches Nachladen von Programmen
  - zeitscheibengesteuerter Scheduler
  - C/C++ API zur Steuerung der Aktoren und zum Zugriff auf Sensordaten
- Infos
  - LegOS Homepage: **`http://www.noga.de/legOS/`**
  - LegOS SF Projektseite: **`http://legos.sourceforge.net/`**
- Nachfolge Projekt: BrickOS
  - BrickOS SF Projektseite: **`http://brickos.sourceforge.net/`**

K.5 Environment

- nach dem Starten des RCX
  - wird ein Programm im ROM ausgeführt
  - kamm man über die IR-Schnittstelle eine Firmware ins RCX laden
- Je nach Firmware lassen sich dann Benutzerprogramme nachladen
- Eigene Programme erstellen
  - Cross-Compiler für H8/300 unter **`/proj/i4vs/pub/cross-gcc-3.2.2/`**
  - gepatchtes LegOS 0.2.5 mit liblnp und lnpd **`/proj/i4vs/pub/legos-0.2.5`**

K.6 LegOS API - Display

- Display löschen

```
void cls ();
```

  - löscht den "Benutzerbereich" des Displays (die ersten 4 Stellen)
- String anzeigen

```
void cputs (char *s);
```

  - es werden nur die ersten 5 Zeichen angezeigt
  - enthält **s** weniger als 5 Zeichen, so werden die übrigen Stellen gelöscht das gilt nicht für die Stelle 0!
- Zahl anzeigen

```
void cputw (unsigned word);   // hexadezimal (0 - 0xffff)
void lcd_int(int value);      // dezimal (-9999 - 0000)
void lcd_unsigned(int value);// dezimal (0 - 9999)

void lcd_digit(int i);        // an letzter Stelle (0 - 9)
```

  - es werden nur die ersten 4 Stellen verwendet (aussnahme lcd_digit)

K.6 LegOS API - Display (2)

- Segmente einzeln ansteuern
  - Segmente abschalten

```
void lcd_hide(char mask);
```

  - Segmente anschalten

```
void lcd_show(char mask);
```

- weitere Infos
  - **`http://legos.sourceforge.net/docs/CommandRef.html`**
  - Quellen von LegOS

K.6 LegOS API - IR Kommunikation

- LegOS Network Protokoll LNP
  - Adressierung zur Identifikation des Kommunikationspartners (z.B. Task)
  - API zum senden und empfangen über die IR-Schnittstelle
- Anwendungsentwicklung auf Hostseite
  - Bibliothek zur Erstellung von Anwendungen auf Hostseite
  - gleiche API wie bei LegOS
  - Daemon (lnpd) als Dispatcher
- Kommunikationssystem auf dem RCX initialisieren
  - unnötig. aber evtl die Reichweite einstellen

```
#include <lnp/lnp-logical.h>
void lnp_logical_range(int far);   // 0: short, 1: long range
```

K.6 LegOS API - IR Kommunikation (2)

- Kommunikationssystem auf Hostseite initialisieren
  - beim **lnpd** anmelden

```
#include <liblnp.h>
lnp_init_result lnp_init(char *tcp_hostname,
                         unsigned short tcp_port,
                         unsigned char lnp_address,
                         unsigned char lnp_mask,
                         int flags);
```

  - **tcp_hostname**, **tcp_port**: IP-Adresse und TCP Port des Rechners auf dem der **lnpd** läuft (0 = localhost, Standartport7776)
  - **lnp_address**: LNP Hostadresse (0 = 0x80)
  - **lnp_mask**: LNP Host Maske (0 = 0xF0)
  - **flags**: z.B. LNP_DISCARD_WHILE_TX
  - Rückgabe bei Erfolg: 0
  - beim lnpd abmelden

```
void lnp_shutdown(void);
```


K.6 LegOS API - IR Kommunikation (3)

- senden
  - direkt (ohne Adressierung)

```
#include <lnp/lnp.h>

int lnp_integrity_write( const unsigned char *data,
                         unsigned char length);
```

  - an eine bestimmte Adresse

```
int lnp_addressing_write( const unsigned char *data,
                          unsigned char length,
                          unsigned char dest,
                          unsigned char srcport);
```

  - bei Erfolg wird 0 zurückgeliefert

K.6 LegOS API - IR Kommunikation (4)

- Empfangs-Handler installieren

```
#include <lnp/lnp.h>

void lnp_integrity_set_handler(lnp_integrity_handler_t
handler);

void lnp_addressing_set_handler(unsigned char port,
                                lnp_addressing_handler_t handler);
```

  - **lnp_integrity_handler_t**:

```
void (*) ( const unsigned char *data,
           unsigned char length);
```

  - **lnp_addressing_handler_t**:

```
void (*) ( const unsigned char *data,
           unsigned char length,
           unsigned char src_address);
```

  - Makros: **LNP_DUMMY_INTEGRITY**, **LNP_DUMMY_ADDRESSING**

K.6 LegOS API - Semaphore

- Semaphore anlegen

```
#include <semaphore.h>
int sem_init (sem_t * sem, int pshared, unsigned int value)
```

  - **pshared** wird ignoriert
- P - Operation

```
int sem_wait (sem_t * sem);
```

  - blockiert bis die Semaphore "frei" ist [`while (sem == 0); sem--;`]
- V- Operation

```
int sem_post (sem_t * sem);
```

- weitere Operationen:

```
int sem_trywait (sem_t *sem);
int sem_getvalue (sem_t *sem, int *sval);
int sem_destroy (sem_t *sem);
```

- Beispiel?

K.6 LegOS API - Taskmanagement

- neuer Task erzeugen

```
pid_t execi ( int(* code_start)(int, char **),
              int argc, char **argv,
              priority_t priority, size_t stack_size);
```

  - ◆ Priorität zwischen 1 und 20 (**PRIO_LOWEST, PRIO_NORMAL PRIO_HIGHEST**)
  - ◆ Stackgröße in Bytes (**DEFAULT_STACK_SIZE** entspricht 512)

- Task beenden (von aussen)

```
void kill (pid_t pid);
```

- weitere Funktionen:

```
void yield (void);
void killall (priority_t p)
```

- Beispiel

```
int code1(int argc, char *argv[]){ ... }
...
int pid1 = execi(code1, 0, (char**)NULL,
                 PRIO_NORMAL, DEFAULT_STACK_SIZE);
```


K.6 LegOS API - Warten

- eine bestimmte Zeit lange warten:

```
unsigned int sleep (unsigned int sec);
unsigned int msleep (unsigned int msec);
```

  - ◆ Rückgabe: Anzahl der verbleibenden (Milli-)Sekunden (oder 0)

- Auf ein Ereignis warten

```
wakeup_t wait_event ( wakeup_t(* wakeup)(wakeup_t),
                      wakeup_t data);
```

  - ◆ Das "Ereignis" ist in der Funktion **wakeup** definiert.
  - ◆ Der Task wird blockiert solange **wakeup(data) 0** zurückliefert.
  - ◆ Beispiel: Auf Tastendruck warten

```
wakeup_t waitkey (wakeup_t w){
    static int i = 0; return i++==w;
}
...
cputs("wait");
wait_event(waitkey, 0);
cputs("ok");
```


K.6 LegOS API - Buttons

- warten bis einer der gewünschten Tasten gedrückt wurde

```
wakeup_t dkey_pressed (wakeup_t data)
```

- warten bis einer der gewünschten Tasten losgelassen wurde

```
wakeup_t dkey_released (wakeup_t data)
```

  - ◆ Keymakros: **KEY_ONOFF**, **KEY_PRGM**, **KEY_RUN**, **KEY_VIEW**, **KEY_ANY**

- auf beliebigen Tastendruck warten

```
int getchar (void)

//Implementierung:
int getchar(void) {
    wait_event(dkey_released,KEY_ANY);
    wait_event(dkey_pressed ,KEY_ANY);
    return dkey;
}
```


K.6 LegOS API - Aktoren (Motor)

- Richtung festlegen:

```
void motor_a_dir(MotorDirection dir);
void motor_b_dir(MotorDirection dir);
void motor_c_dir(MotorDirection dir);
```

- Geschwindigkeit festlegen:

```
void motor_a_speed(int speed);
void motor_b_speed(int speed);
void motor_c_speed(int speed);
```

  - ◆ wobei folgende Makros definiert sind **MIN_SPEED**, **MAX_SPEED**

K.6 LegOS C++ API - Aktoren (Motor)

- Klasse Motor

```
#include <c++/Motor.H>;

class Motor {
public:
    enum Port { A, B, C };
    enum Limits { min = 0, max = 255 };
    Motor(const Port port);

    void speed(const unsigned char speed);
    void direction(MotorDirection dir);

    void Forward();
    void Forward(unsigned char s);
    void Reverse();
    void Reverse(unsigned char s);
    void Brake();
    void Brake(int duration);
    void Off();
}

typedef enum {off, fwd, rev, brake} MotorDirection;
```


K.6 LegOS C++ API - Sensoren

- siehe C++ Header von legOS (**`include/c++/*.H`**)
- dort stehen eine Reihe von Klassen zur Abfrage der einzelnen Sensortypen zur Verfügung.

K.6 Aufgabe 7

- RPC-System auf RCX portieren
- Hostanwendung dient als Fernsteuerung