```
/**
 * Initialisiert alle wichtigen Module.
 *
 * Initializes all important modules.
 */
void Init(void);

/**
 * Fährt eine bestimmte Strecke ODER dreht mit vorgegebenem Winkel
 * unter Berücksichtigung der vorgegebenen Geschwindigkeit.
 *
 * Nachdem die/der vorgegebene Strecke/Winkel zurückgelegt wurde, schaltet
 * GoTurn() die Motorgeschwindigkeit auf 0, ohne dass die Richtungsvorgabe
 * modifiziert wird.
 *
 * Während GoTurn() aktiv ist, darf EncoderPoll() nicht aufgerufen werden (etwa
 * durch ISRs), die Auswertung der Odometriedaten erfolgt automatisch.
 *
 * Drives a specified distance OR turns by a given angle, with respect to a
 * defined speed value.
 *
 * The motor speed is set to 0 by GoTurn() after driving/turning the specified
 * distance/angle, without changing the configured motor directions.
 *
 * EncoderPoll() must not be called while GoTurn() is active (for example from
 * an interrupt context), the encoders will be monitored by GoTurn() itself.
 */
extern void GoTurn(int16_t distance, int16_t degree, uint8_t speed);


/**
 * Schaltet den Status der Back-LEDs um, sollen eine oder beide Back-LEDs
 * aktiviert werden, wird automatisch von Odometrie auf die Back-LEDs umgestellt.
 *
 * Diese Funktion existiert nur aus Kompatibilitätsgründen, intern wird
 * BackLEDFast() aufgerufen.
 *
 * Zur Auswahl stehen für jede LED die Zustände ON und OFF
 *
 * Configures the back LEDs according to the passed states. The back/odometry
 * LEDs are automatically configured to back LED mode if at least one back LED
 * shall be enabled.
 *
 * This function does only exist for compatibility reasons, BackLEDFast() is
 * used internally.
 *
 * Each LED might be configured as ON or OFF.
 */
extern void BackLED(const uint8_t left, const uint8_t right);

/**
 * Schaltet den Status der Front/Linienfolge-LED um.
 *
 * Configures the front/line following LED according to the passed state.
 */
extern void FrontLED(const uint8_t status);

/**
 * Setzt die Status-LED auf die übergebene Farbe oder schaltet sie aus.
 *
 * Configures the status LED according to the passed color/state.
 */
extern void StatusLED(const uint8_t color);
```

```c
/**
 * Setzt die Fahrtrichtung der beiden Motoren.
 *
 * Als Fahrtrichtung/-Modus stehen zur Auswahl: FWD, RWD/BWD, BREAK und FREE.
 *
 * Configures the driving direction for both motors.
 *
 * Available choices are: FWD, RWD/BWD, BREAK and FREE.
 */
extern void MotorDir(const uint8_t left_dir, const uint8_t right_dir);

/**
 * Setzt die Fahrtgeschwindigkeit beider Motoren.
 *
 * Laut ASURO-Manual bewegt sich der ASURO ab Werten von ca. 60, der Wert
 * UINT8_MAX (255) entspricht der Höchstgeschwindigkeit.
 *
 * Configures the driving speed for both motors.
 *
 * According to the ASURO manual, the ASURO starts to move for speed values of
 * about 60, the value UINT8_MAX (255) represents the top speed.
 */
extern void MotorSpeed(const uint8_t left_speed, const uint8_t right_speed);

/**
 * Erfasst die Werte der Liniensensoren synchron (blockierend).
 *
 * Captures a measurment of the line sensors in a synchronous (blocking) manner.
 */
extern void LineData(uint16_t * const data);

/**
 * Erfasst den genauen Tasterstatus und gibt diesen zurück.
 *
 * Zur bequemeren Auswertung der Taster können die Makros SWITCH_1 bis SWITCH_6
 * verwendet werden.
 *
 * Reads/returns the state of all bumper switches fused into a single byte.
 *
 * The macros SWITCH_1 to SWITCH_6 are provided for a convenient access of the
 * bits representing the according bumper switch.
 *
 * The switch interrupt observation feature should be disabled if PollSwitch()
 * is called, otherwise the readout process will trigger the switch interrupt.
 */
extern uint8_t PollSwitch(void);

/**
 * Verhält sich wie sleep(), verwendet als Zeitbasis aber Millisekunden
 *
 * Behaves like sleep() but uses milliseconds as timebase.
 */
void msleep(uint16_t ms);
```