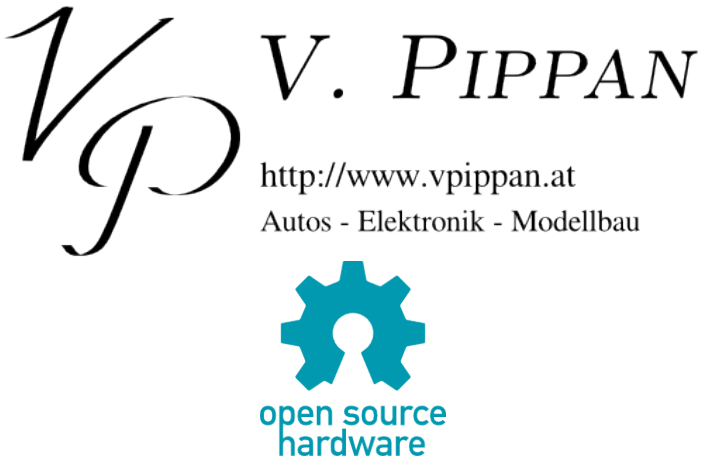
ATmega8 GPS Logger Basismodul

(Rev B)

V. Pippan
http://www.vpippan.at

26. Oktober 2014



- Bitcoin: 1KdFDDxe7rc32ccWjH6uwrMbk9vG5sQVKW
- Flattr: flattr.com/t/1164322

Inhaltsverzeichnis

1. Allgemeines

Seit meinem Maturaprojekt bin ich fasziniert davon GPS Daten aufzuzeichnen und anzusehen. Mit den neuen Möglichkeiten von Openstreetmap und GoogleEarth wurde mein Wunsch nach einem funktionierendem GPS Logger immer größer. Die käuflichen Geräte waren mir alle zu teuer, vorallem wenn ich mir sowas selbst bauen kann. Außerdem wollte ich schon länger mal wieder etwas mit Mikrocontrollern machen, somit war dieses Projekt der ideale Einstieg für mich. Als Mikrocontroller wählte ich dann einen Typ von Atmel, hauptsächlich weil es für die Atmel Controller eine sehr gute Linux Unterstützung gibt und weil diese bei Bastlern weit verbreitet sind.

Falls größeres Interesse an dieser Schaltung besteht, bin ich gerne bereit mehrere Platinen fertigen zu lassen und die Schaltung als Bausatz abzugeben.

Für Fragen, Anregungen, Erfahrungsaustausch, Probleme, Beschwerden, etc. oder um über das Projekt zu diskutieren, schreiben Sie doch in mein Forum oder eine E-Mail (webmaster@vpippan.at) direkt an mich.

Für dieses Projekt suche ich jemanden, der Lust hat an der Microcontroller Programmierung (ATmega8 in C) mitzuarbeiten. Aber natürlich ist auch jede Hilfe in anderer Form (Hardware Entwurf, Latex Dokumentation, etc.) willkommen. Falls Sie Interesse haben kontaktieren Sie mich bitte, dann erhalten Sie Zugriff auf das Subversion Projektarchiv.

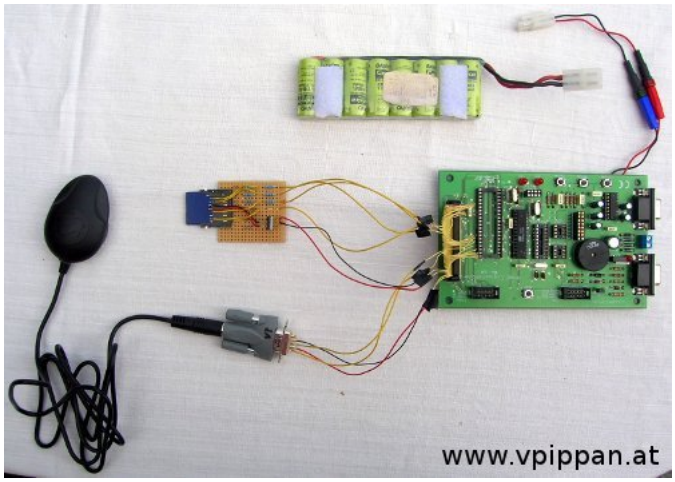
Die Downloads für die Hardware und Software zu diesem Projekt finden Sie auf der Projektseite: `http://www.vpippan.at/index.php?pid=Elektronik_Projekte_GpsLogger`.

2. Hardwarebeschreibung

2.1. Allgemeines

Das Herzstück der Schaltung ist ein Mikrocontroller ATmega8. Die von der GPS Maus (Navilock NL-303P) erzeugten seriellen Daten werden mit einem MAX233 auf TTL Pegel gebracht und anschließend vom µC auf einer SD Karte gespeichert. Für die SD Karte steht ein eigener 3,3 V Spannungsregler zur Verfügung, da der µC und die GPS Maus mit 5 V betrieben werden, die SD Karte aber nur 3,3 V verträgt. Die Pegelumsetzung des SPI Signals von 5 V auf 3,3 V (für die SD Karte) erfolgt über einen einfachen Spannungsteiler. Nicht gerade die beste Lösung, aber sie funktioniert. Ich habe den Spannungsteiler gewählt, da er zu diesem Zeitpunkt die einfachste und vorallem schnellste Lösung für mich war.

2.2. Prototyp



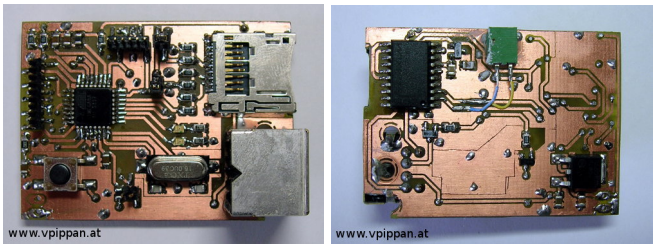
Die Stromaufnahme der Schaltung beträgt 100 mA (bei Datenaufzeichnung), was mit dem eingebauten Akku für 8 Stunden Aufzeichnung ausreicht. Die verwendete SD Karte ist ebenfalls groß genug um diese Aufzeichnungsdauer zu ermöglich.

2.3. Rev A

Das Gehäuse der hier abgebildeten (Bild 2.3) Version hat eine Größe von $(170 \times 85 \times 34)$ mm (L x B x H) und mit einem Gewicht von ca. 200 g kann der ganze Logger auch einfach mitgenommen werden.

2.4. Rev B

Bei dieser neuen Version des Loggers handelt es sich um ein komplett neues Design der Schaltung. Zur Verkleinerung des gesamten Aufbaus sind nur SMD Bauteile verwendet worden. Schaltungstechnisch sind einige Verbesserungen aus Atmel App Notes und Erfahrungen eingeflossen.



2.4.1. Technische Daten

- Abmessungen der Platine: ca. (51×37) mm
- Masse: 20 g Basismodul / 70 g GPS Maus (mit Kabel und Gehäuse) / 50 g LiPo Akku (2S 910 mA h)
- komplett in SMD Technik aufgebaut, Bauteile in 0805 Größe
- Leiterbahnbreiten 0,4 mm / Leiterbahnabstände 0,3 mm
- Durchkontaktierung außen 1 mm und Bohrung 0,5 mm
- getrennte Masseflächen für Quarz, Analog- und Digitalteil
- Spannungsversorgung 5,7 V bis 10 V
- Überwachung der Akku Spannung → GPS Modul und Versorgung des Zusatzboard bei leerem Akku vom µC abschaltbar

2.5. Mögliche Änderungen für eine neue Version

- Richtiger Pegelumsetzer für die SD Karte
- Mini-DIN Stecker (hat sich als sehr unpraktisch erwiesen) der GPS Maus durch einen kleineren (Pinleiste?) ersetzen
- Diode für Verpolungsschutz beim Akku und Sicherung

2.6. Mögliche Erweiterungsmodule

Um schnell und einfach Erweiterungsmodule zu entwickeln gibt es eine Schaltplan / PCB Layout Vorlage für das Erweiterungsmodul. Diese findet sich im Downloadbereich der Projektseite und enthält alle notwendigen Verschaltungen sowie die korrekte Platzierung der Steckverbinder.

- Display Modul für die Live anzeige der Daten → Verwendung des Loggers für Geocaching als normales GPS Gerät
- Erweiterungsmodul mit ansteckbaren Sensoren (Drehzahl, Temperatur, Beschleunigung, etc.) → für den Einsatz im Modellbau
- Erweiterungsmodul nur mit Beschleunigungssensor
- Erweiterungsmodul für Kommunikation mit dem Jeti Duplex System
- Erweiterungsmodul mit direkt integriertem TTL GPS Modul statt einer ansteckbaren GPS Maus → kompakter, weniger Platzbedarf und Gewicht

Eine automatische Unterscheidung der Zusatzmodule kann erreicht werden, indem bei jedem Zusatzmodul ein anderer (nicht verwendeter) Pin des ADC auf „HIGH“ gelegt wird. Dadurch kann die Software das angeschlossene Zusatzmodul erkennen und die notwendigen Funktionen können in der Software integriert sein → es wird nicht für jedes Zusatzmodul eine eigene Software benötigt.

Bei Verwendung eines Moduls mit LCD Anzeige wäre es möglich, das LCD in ein seperates Gehäuse einzubauen. Dann kann das Basismodul z. B. im Rucksack getragen werden und das LCD läßt sich an einer anderen, gut sichtbaren Stelle, befestigen.

2.7. Fertigungsunterlagen

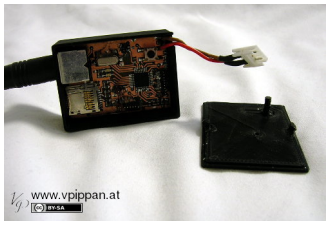
Alle benötigten Informationen, zum Aufbau des Moduls und Vorlagen für den eigenen Entwurf von Zusatzmodulen, finden sich im Anhang A.

2.8. Gehäuse

Extra für diesen GPS Logger habe ich mit Hilfe eines 3D CAD Programms ein passendes Gehäuse entworfen. Es stehen zwei Varianten zur Verfügung:

- Variante mit externer Batterie (z. B. für den Einsatz im Modellbau → Verwendung einer bereits im Modell vorhandenen Batterie)
  - Masse: 11 g (3D gedrucktes ABS)
  - Abmessungen: $(53 \times 40 \times 22)$ mm (L x B x H)
- Variante mit interner Batterie (Die Batterie und zusätzlich benötigte Schalter und Ladebuchsen finden im Gehäuse Platz → Stand-Allone Verwendung des Loggers)
  - Abmessungen: $(124 \times 40 \times 22)$ mm (L x B x H)

Der Gehäusedeckel wird über zwei Haken im Gehäuse eingeklipst und kann daher auch einfach ohne Werkzeug entfernt werden. Die CAD Daten sind in der KiCAD Projekt Daten Datei (siehe Downloadbereich auf der Projektwebseite) zu finden. Somit kann man das Gehäuse leicht auf eigene Befürfnisse anpassen und nachbauen oder selber auf einem 3D Drucker ausdrucken.



3. Berechnungen

3.1. Max. Strom durch den Spannungsregler

$$P_{D,max} = (V_{ein,max} - V_{aus,min}) \cdot I_{aus,max} + V_{ein,max} \cdot I_{GND} \tag{1}$$

$$R_{\vartheta,JA} = \frac{150\,°\mathrm{C} - T_A}{P_D} \tag{2}$$

Folgende Werte werden angenommen:

- $\mathrm{V_{ein,max}} = 8{,}4\,\mathrm{V}$ (maximale Akkuspannung)
- $\mathrm{V_{aus,min}} = 4{,}755\,\mathrm{V}$ (aus dem Datenblatt)
- $\mathrm{I_{aus,max}} = 0{,}5\,\mathrm{A}$
- $\mathrm{I_{GND}} = 0{,}02\,\mathrm{A}$ (aus Datenblatt)
- $\mathrm{T_A} = 35\,°\mathrm{C}$ (angenommene Umgebungstemperatur am Einbauort der Schaltung)

Mit diesen Werten erhält man aus der Gleichung 1: $\mathrm{P_D} = 2\,\mathrm{W}$ und aus Gleichung 2: $\mathrm{R_{\vartheta,JA}} = 57{,}5\,\frac{°\mathrm{C}}{\mathrm{W}}$.

Um den Spannungsregler innerhalb der zulässigen Temperaturen zu halten, muß der berechnete Wert von $\mathrm{R_{\vartheta,JA}}$ größer als $\mathrm{R_{\vartheta,JA}} = 60\,\frac{°\mathrm{C}}{\mathrm{W}}$ für das Gehäuse (diesen Wert bekommt man aus dem Datenblatt) sein!

Wie man sieht ist das bei meiner Schaltung knapp nicht der Fall. Allerdings ist der Spannungsregler mit dem GND Pin ebenfalls an der Platine angelötet und wird daher über diesen Pfad zusätzlich gekühlt (die Platine wirkt wie ein Kühlkörper). Aus diesem Grund sollte die Schaltung bei den oben gegebenen Bedingungen noch problemlos funktionieren.
Für die Berechnung habe ich die Maximal und Minimal Werte aus dem Datenblatt verwendet. Führt man die Berechnung mit den typischen Werten durch, dann sieht das Ergebnis besser aus, es enthält aber keine Sicherheiten mehr.

Will man die Schaltung bei höheren Eingangsspannungen $V_{ein,max}$ und/oder höheren Umgebungstemperaturen T_A mit dem maximalen Strom $I_{aus,max} = 0{,}5\,A$ betreiben, dann ist am Spannungsregler unbedingt ein Kühlkörper zu montieren! Die dafür benötigte Berechnung ist im Datenblatt gegeben.

3.2. Max. Strom durch die MOSFETs

Laut Datenblatt ist der maximale Drainstrom $I_D = 450\,mA$ bei $85\,°C$ Umgebungstemperatur und $I_D = 630\,mA$ bei $25\,°C$ Umgebungstemperatur. Bei meiner Schaltung sollten 400 mA pro Transistor nie überschritten werden. Der Spannungsregler kann maximal 500 mA liefern (siehe 3.1) und die GPS Maus + Basismodul benötigt alleine schon ca. 100 mA. Daher können die 400 mA an T2 (Abschaltung Zusatzmodul) auch nie überschritten werden.

3.3. To-Do

- Berechnung Leiterbahnbreiten
- Einfluss vom Strom in den ADC auf den Spannungsteiler → hochohmiger Spannungsteiler mit hochohmiger Last → Verfälschung des gemessenen Spannungswertes?

4. Messungen

To-Do

5. Softwarebeschreibung

5.1. Allgemeines

Die Software ist in C geschrieben und steht unter der GPLv3. Die aktuelle Version ist geeignet für die Hardware Versionen Rev. A und Rev. B.

An dieser Stelle einen großen Dank an Martin Matysiak von dem die Software stammt (mit kleinen Änderungen von mir).

5.2. Eigenschaften der Software

To-Do

5.3. Softwaredokumentation

Eine vollständige Dokumentation der Software findet sich im Anhang B oder als HTML Version auf meiner Website.

6. Links

- gLogger von Martin Matysiak
- gLogger Mini von Martin Matysiak
- gpsbabel.org - Freie Software für GPS Datenumwandlung und Übertragung

- DataExplorer - Freie Software um Informationen (Höhe, Geschwindigkeit, etc.) aus den NMEA Daten zu extrahieren
- Marble - Virtueller Globus und Welt Atlas

7. Versionsgeschichte

7.1. Hardware Basismodul (Rev A)

- Ursprünglich veröffentlichte Version mit Bauteilen für Durcksteckmontage

7.2. Hardware Basismodul (Rev B – 20130624)

- Komplettes Neudesign der Platine mit SMD Bauteilen
- MAX 232 durch MAX233A ersetzt (benötigt keine Kondensatoren)
- Versorgungen vom GPS Modul und vom Zusatzboard sind jetzt komplett abschaltbar
- Allgemeine Verbesserung des Schaltungsdesigns

7.3. Hardware Basismodul (Rev B – 20130809)

- Unnötige Durchkontaktierung entfernt
- Update der Fertigungsunterlagen

7.4. Software Basismodul (20131018)

- Ursprünglich veröffentlichte Version (fast identisch mit der gLogger Software von Martin Matysiak)

7.5. Dokumentation Basismodul (20131112)

- Berechnung für Spannungsregler hinzugefügt, maximale Eingangsspannung im Schaltplan geändert
- Kontrolliert ob MOSFETs für die auftretenden Ströme geeignet sind

7.6. Dokumentation Basismodul (20140622)

- Dokumentation mit LaTeX geschrieben anstatt eines Spreadsheets

7.7. Hardware Zusatzmodul Vorlage (Rev B – 20140826)

- Bohrung eingefügt, damit Status LEDs nicht durch das Zusatzmodul verdeckt werden
- Bemaßungen eingefügt und verbessert

7.8. Dokumentation Basismodul (20141026)

- Dokumentation für Zusatzmodul eingefügt
- Gehäuse erwähnt und Bilder eingefügt
- Fehler in Tabellenbeschriftung korrigiert
- In der Einkaufsliste Links zu den Bauteilen eingefügt

8. Gewährleistungsausschluß

Es besteht keinerlei Gewährleistung für das Projekt, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Urheberrechtsinhaber und/oder Dritte das Projekt so zur Verfügung, „wie es ist“, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich – aber nicht begrenzt auf – die implizite Gewährleistung der Marktreife oder der Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Projekts liegt bei Ihnen. Sollte sich das Projekt als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

9. Haftungsbegrenzung

In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Urheberrechtsinhaber oder irgendein Dritter, der das Projekt wie oben erlaubt modifiziert oder übertragen hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Projekts oder der Unbenutzbarkeit des Projekts folgen (einschließlich – aber nicht beschränkt auf – Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen, oder dem Unvermögen des Projekts, mit irgendeinem anderen Projekt zusammenzuarbeiten), selbst wenn ein Urheberrechtsinhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.

10. Interpretation von 8 und 9

Sollten der o.a. Gewährleistungsausschluß und die o.a. Haftungsbegrenzung aufgrund ihrer Bedingungen gemäß lokalem Recht unwirksam sein, sollen Bewertungsgerichte dasjenige lokale Recht anwenden, das einer absoluten Aufhe-

bung jeglicher zivilen Haftung in Zusammenhang mit dem Projekt am nächsten kommt, es sei denn, dem Projekt lag eine entgeltliche Garantieerklärung oder Haftungsübernahme bei.

11. Dokumentation To-Do

- Bedienugsanleitung (Entwicklungsumgebung aufsetzen für Software und Hardware, µC Software installieren, Anschluss des Moduls, etc.)
- Schriftart vom V. Pippan Schriftzug überall ändern (Kapitälchen und kursiv)

A. Fertigungsunterlagen

A.1. Einkaufs Liste

Bauteil	Wert	Hersteller	Farnell Best. Nr.	Herstellerbezeichnung	Stückpreis/€	Anzahl	Preis/€
Kapazität	27 pF	Multicomp	1759196	MCCA000323	0,013	2	0,026
Kapazität	100 nF	Multicomp	1759167	MCCA000296	0,01	4	0,04
Kapazität	10 µF	TDK	1907346	C2012Y5V1C106Z	0,148	1	0,148
Kapazität	4,7 µF	Multicomp	1759478	MCCA000595	0,012	1	0,012
Kapazität	330 nF	Multicomp	1759174	MCCA000303	0,018	1	0,018
Kapazität	1 µF	Multicomp	1759479	MCCA000596	0,044	1	0,044
Stiftleiste	10 polig	Harwin	1099570	M52-040000P1045	1,33	2	2,66
Buchse	MINIDIN-6_PS/2	TE Connectivity	1863504	5749180-1	1,06	1	1,06
µ C	ATmega8A-AU	Atmel	1748532	ATMEGA8A-AU	2,89	1	2,89
Schnittstellentreiber	MAX233A-SO	Maxim	2113687	MAX233ACWP+G36	9,81	1	9,81
Induktivität	10 µH	Taiyo Yuden	2112893	LB2012T100M	0,084	2	0,168
LED	Grün	Kingbright	2099239	KPT-2012SGC	0,08	1	0,08
LED	Rot	Kingbright	2099236	KPT-2012EC	0,08	1	0,08
Quarz	16 MHz	TXC	1842293	9C-16.000MAAJ-T	0,421	1	0,421
Widerstand	10 kΩ	Yageo (Phycomp)	9237755	RC0805FR-0710KL	0,006	3	0,018
Widerstand	270 Ω	Yageo (Phycomp)	9237410	RC0805FR-07270RL	0,006	2	0,012
Widerstand	33 kΩ	Yageo (Phycomp)	9237810	RC0805FR-0733KL	0,006	1	0,006
Widerstand	1,8 kΩ	Yageo (Phycomp)	9237526	RC0805FR-071K8L	0,006	3	0,018
Widerstand	3,3 kΩ	Yageo (Phycomp)	9237682	RC0805FR-073K3L	0,006	3	0,018
Widerstand	100 Ω	Yageo (Phycomp)	9237364	RC0805FR-07100RL	0,006	2	0,012
Taster	Start/Stop	TE Connectivity	3801305	FSM4JSMA	0,327	1	0,327
SD Karten Sockel	Micro SD Socket	Molex	2064063	502774-0891	1,81	1	1,81
MOSFET	DMN2004K-7	Diodes Inc.	1713842	DMN2004K-7	0,116	2	0,232
Spg. Regler	NCP5501DT50G	ON Semiconductor	1703369	NCP5501DT50G	0,725	1	0,725
Spg. Regler	LM3480IM3-3.3	National Semiconductor	1469102	LM3480IM3-3.3/NOPB	1,15	1	1,15
PCB		V. Pippan		GL_B_PCB_RevB	10	1	10

Der Gesamtpreis mit allen notwendigen Bauteilen ergibt sich somit zu 31,80€.

Falls die Bauteile nicht bereits vorhanden sind und die Mindestbestellmengen bestellt werden müssen, dann ergeben sich die folgenden Preise:

Bauteil	Wert	Min. Bestellmenge	Preis (Mindestbestellung)/€
Kapazität	27 pF	100	1,3
Kapazität	100 nF	100	1
Kapazität	10 µF	50	7,4
Kapazität	4,7 µF	100	1,2
Kapazität	330 nF	100	1,8
Kapazität	1 µF	100	4,4
Stiftleiste		1	2,66
Buchse	MINIDIN-6_PS/2	10	10,6
µ C	ATmega8A-AU	1	2,89
Schnittstellentreiber	MAX233A-SO	1	9,81
Induktivität	10 µH	1	0,168
LED	Grün	1	0,08
LED	Rot	1	0,08
Quarz	16 MHz	1	0,421
Widerstand	10 kΩ	50	0,3
Widerstand	270 Ω	50	0,3
Widerstand	33 kΩ	50	0,3
Widerstand	1,8 kΩ	50	0,3
Widerstand	3,3 kΩ	50	0,3
Widerstand	100 Ω	50	0,3
Taster	Start/Stop	25	8,175
SD Karten Sockel	Micro SD Socket	1	1,81
MOSFET	DMN2004K-7	5	0,58
Spg. Regler	NCP5501DT50G	1	0,725
Spg. Regler	LM3480IM3-3.3	1	1,15
PCB		1	10

Womit sich der Gesamtpreis zu 68,10€ für alle Bauteile ergibt.

Bestellnummern, Mindestbestellmengen und Stückpreise stammen von Farnell Österreich.

Mit den Fertigungsunterlagen zum Herunterladen können Sie entweder die Platine selber ätzen oder von einem Auftragsfertiger herstellen lassen.

Zusätzlich können Sie auch eine Platine bei mir bestellen. Diese wird dann von mir geätzt und gebohrt.

Falls Sie nicht über die Hardware verfügen um den µC selber zu programmieren, dann besteht natürlich auch die Möglichkeit, diesen bereits programmiert bei mir zu kaufen.

Wenn Ihnen die Mindestbestellmengen zu groß sind, z. B. weil Sie nicht so viele Bauteile benötigen, dann melden Sie sich einfach bei mir. Falls ich das gewünschte Bauteil noch auf Lager habe verkaufe ich es Ihnen gerne!

A.1.1. Zusätzlich benötigte Teile

- serielle GPS Maus (zum Beispiel: Navilock GPS Maus, ca. 23€)
- Micro SD Karte
- Gehäuse (siehe 2.8)
  - Bei interner Batterie: Batterie, Schalter und Buchsenleiste
  - Bei externer Batterie: Anschlusskabel

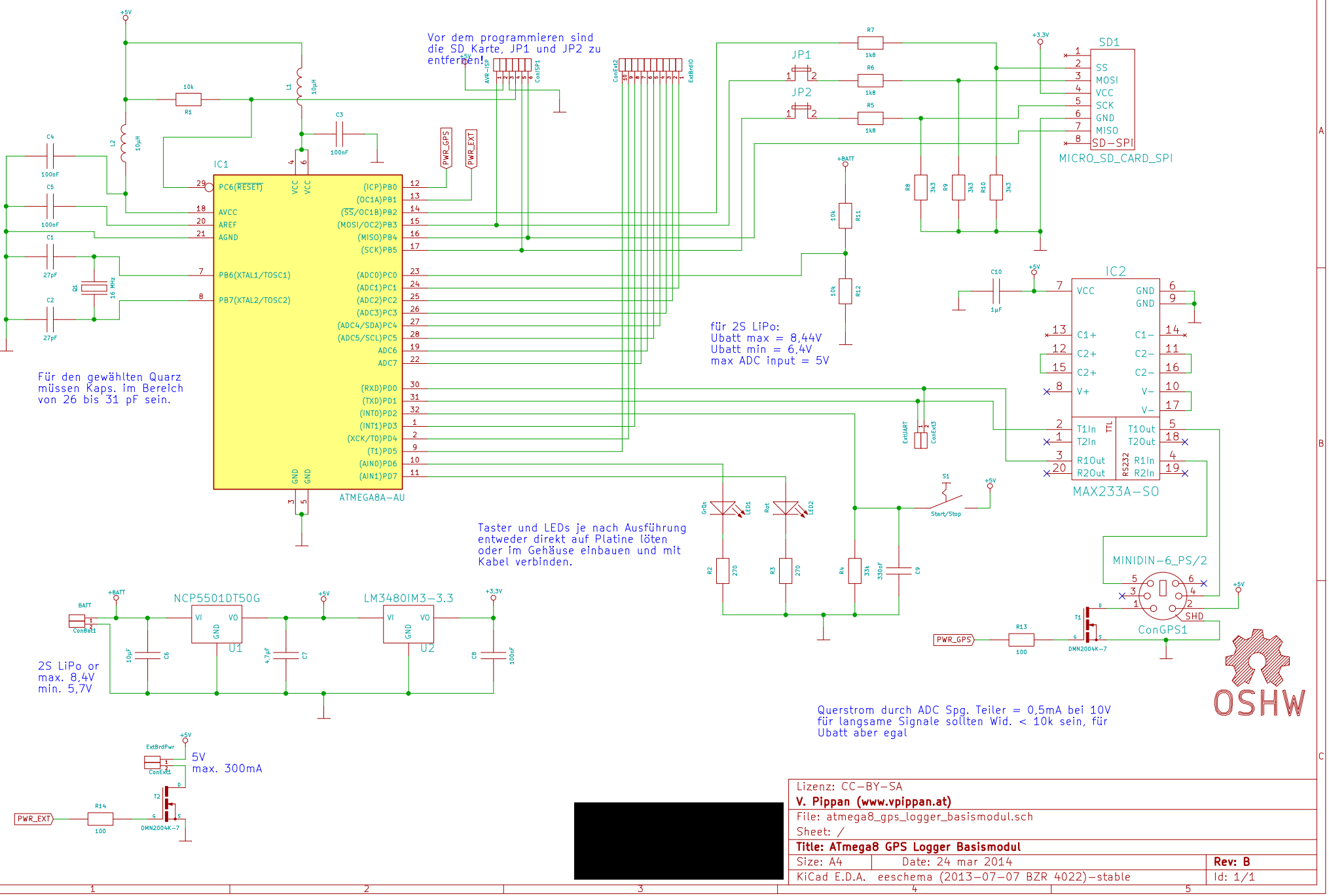
A.2. Bauteil Liste

Referenz	Wert	Footprint	Toleranz	Hersteller	Herstellerbezeichnung
C1	27 pF		±5 %	Multicomp	MCCA000323
C2	27 pF		±5 %	Multicomp	MCCA000323
C3	100 nF		−20 % / 80 %	Multicomp	MCCA000296
C4	100 nF		−20 % / 80 %	Multicomp	MCCA000296
C5	100 nF		−20 % / 80 %	Multicomp	MCCA000296
C6	10 µF		−20 % / 80 %	TDK	C2012Y5V1C106Z
C7	4,7 µF		−20 % / 80 %	Multicomp	MCCA000595
C8	100 nF		−20 % / 80 %	Multicomp	MCCA000296
C9	330 nF		−20 % / 80 %	Multicomp	MCCA000303
C10	1 µF		±10 %	Multicomp	MCCA000596
ConBat1	BATT				
ConExt1	ExtBrdPwr				
ConExt2	ExtBrdIO				
ConExt3	ExtUART				
ConGPS1	MINIDIN-6_PS/2			TE Connectivity	5749180-1
ConISP1	AVR-ISP				
IC1	ATmega8A-AU			Atmel	ATMEGA8A-AU
IC2	MAX233A-SO			Maxim INTEGRATED PRODUCTS	MAX233ACWP+G36
JP1	JUMPER_1				
JP2	JUMPER_1				
L1	10 µH		±20 %	Taiyo Yuden	LB2012T100M
L2	10 µH		±20 %	Taiyo Yuden	LB2012T100M
LED1	Grün			KINGBRIGHT	KPT-2012SGC
LED2	Rot			KINGBRIGHT	KPT-2012EC
Q1	16 MHz		±30ppm	TXC	9C-16.000MAAJ-T
R1	10 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-0710KL
R2	270 Ω		±1 %	Yageo (Phycomp)	RC0805FR-07270RL
R3	270 Ω		±1 %	Yageo (Phycomp)	RC0805FR-07270RL
R4	33 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-0733KL
R5	1,8 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-071K8L
R6	1,8 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-071K8L
R7	1,8 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-071K8L
R8	3,3 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-073K3L
R9	3,3 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-073K3L

Referenz	Wert	Footprint	Toleranz	Hersteller	Herstellerbezeichnung
R10	3,3 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-073K3L
R11	10 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-0710KL
R12	10 kΩ		±1 %	Yageo (Phycomp)	RC0805FR-0710KL
R13	100 Ω		±1 %	Yageo (Phycomp)	RC0805FR-07100RL
R14	100 Ω		±1 %	Yageo (Phycomp)	RC0805FR-07100RL
S1	Start/Stop			TE Connectivity / ALCOSWITCH	FSM4JSMA
SD1	Micro SD Socket			Molex	502774-0891
T1	DMN2004K-7			Diodes Inc.	DMN2004K-7
T2	DMN2004K-7			Diodes Inc.	DMN2004K-7
U1	NCP5501DT50G			ON Semiconductor	NCP5501DT50G
U2	LM3480IM3-3.3			National Semiconductor	LM3480IM3-3.3/NOPB
PCB				V. Pippan	GL_B_PCB_RevB

A.3. Weitere Unterlagen

- Schaltplan
- PCB Größe und Bohrplan
- PCB Belichtungsvorlage
- PCB Bestückungsplan



V. Pippan (www.vpippan.at)

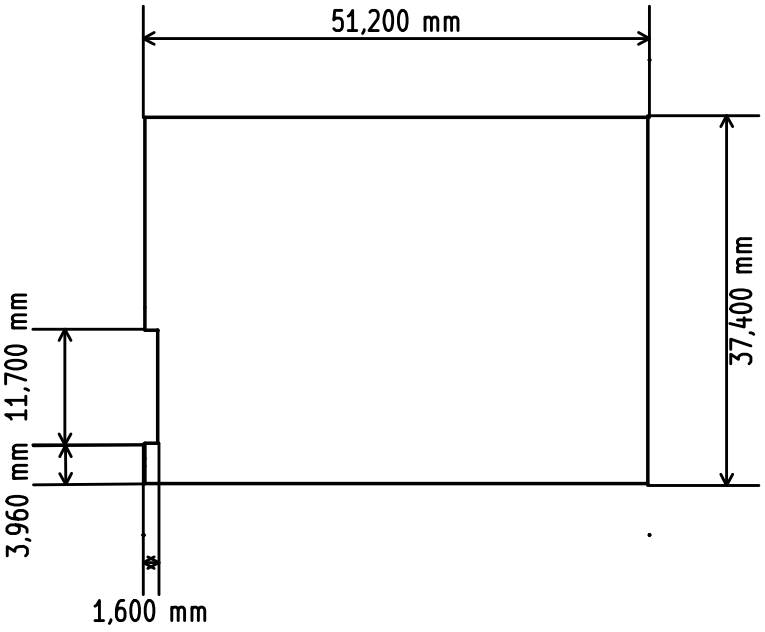
File: atmega8_gps_logger_basismodul.sch

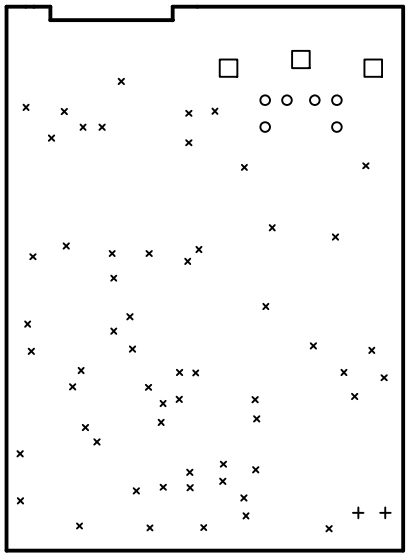
Sheet: /

Title: ATmega8 GPS Logger Basismodul

Size: A4	Date: 24 mar 2014	**Rev: B**
KiCad E.D.A. eeschema (2013-07-07 BZR 4022)-stable		Id: 1/1

ATmega8 GPS Logger Basismodul
V. Pippan (www.vpippan.at)
Rev B

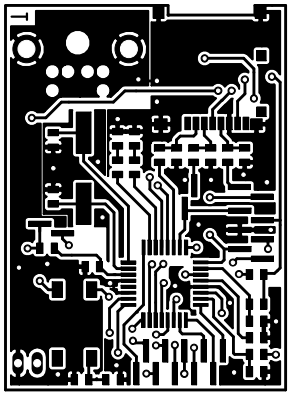


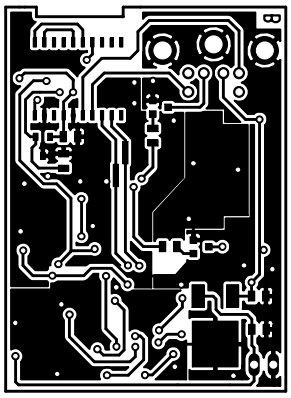


Drill Map:

× 0.50mm / 0.020" (58 holes)
○ 0.90mm / 0.035" (6 holes)
+ 1.00mm / 0.039" (2 holes)
□ 2.30mm / 0.091" (3 holes)

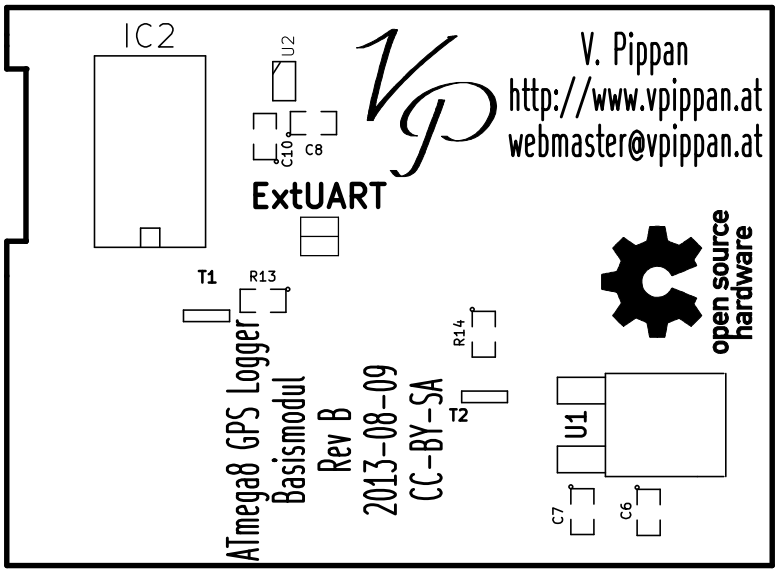
ATmega8 GPS Logger Basismodul
V. Pippan (www.vpippan.at)
Rev B



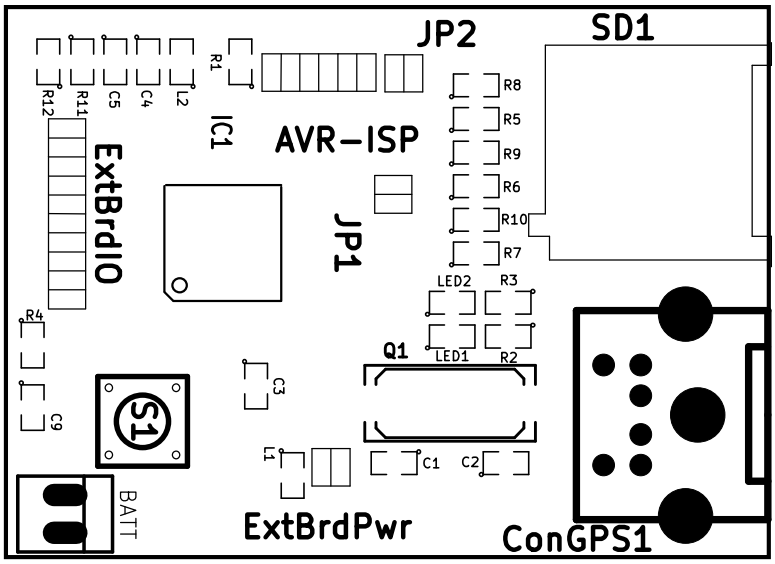


Top -> gespiegelt & von oben betrachtet
Bottom -> nicht gespiegelt & von oben betrachtet

ATmega8 GPS Logger Basismodul
V. Pippan (www.vpippan.at)
Rev B
CC-BY-SA



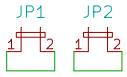
Bestückung Rückseite

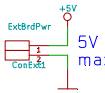


Bestückung Vorderseite

A.4. Vorlage für Zusatzmodule

- Schaltplanvorlage
- PCB Größe und Bohrplan für Zusatzmodule



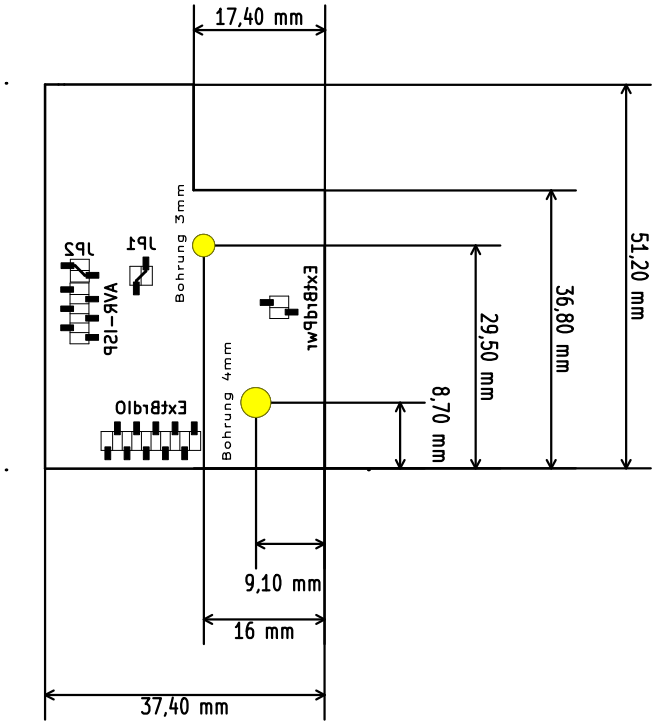


5V
max. 300mA

[redacted]

OSHW

Lizenz: CC-BY-SA		
V. Pippan (www.vpippan.at)		
File: atmega8_gps_logger_zusatzmodul_vorlage.sch		
Sheet: /		
Title: ATmega8 GPS Logger Zusatzmodul Vorlage		
Size: A4	Date: 24 mar 2014	**Rev: B**
KiCad E.D.A. eeschema (2013-07-07 BZR 4022)-stable		Id: 1/1



V. Pippan (www.vpippan.at)

File: atmega8_gps_logger_zusatzmodul_vorlage.kicad_pcb
Sheet: 1/1

Title: ATmega8 GPS Logger Zusatzmodul Vorlage

Size: A4	Date: 26 aug 2014	**Rev: B**
KiCad E.D.A.	pcbnew (2013-07-07 BZR 4022)-stable	Id: 1/1

B. Softwaredokumentation

ATmega8GPSLoggerBasismodul

20131018

Generated by Doxygen 1.8.7

Fri Jun 20 2014 14:56:36

Contents

Chapter 1

Todo List

File atmega8_gps_logger_basismodul.c

Code schöner schreiben, Kommtentare einfügen, Doxygen Doku verbessern

- FAT, UART und MMC Libraries in eigenem Ordner speichern
- In neuer Version auf FAT verzichten und Daten ohne Dateisystem auf SD Karte schreiben (siehe g↩ LoggerMini) -> weniger Speicher für Code, schneller?
- Wartezeit für GPS Datenempfang einstellbar machen (eventuell nur Positionsdaten auslesen?)
- GPS Maus mit anderen Einstellungen initialisieren (höhere Datenrate, andere NMEA Sätze?)
- Start / Stop aus dem File entfernen -> Überprüfen ob Datei vorhanden und dann neue anlegen (Dateinamen bei jedem starten ändern und damit neue Datei anlegen -> String append ?)
- Akkuspannung messen und bei zu niedriger Spannung -> sleep mode (Datei abschließen), GPS und Zusatzmodul abschalten und warnen
- GPS nur einschalten wenn Aufzeichnung aktiviert, sonst ausschalten
- GPS Daten auf Display ausgeben, zusätzliche Sensoren, Jeti Kommunikation -> größerer uC benötigt? -> ATmega32?
- Berechnungen für Oldtimerrallye Training machen und am Display ausgeben
- Erkennung von Zusatzmodulen und Code für diese
- Bei Verwendung mit LCD Zusatzmodul das Logging abschaltbar machen -> nur Anzeige der Daten

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Module Documentation

5.1 UART Library

Interrupt UART library using the built-in UART with transmit and receive circular buffers.

Macros

- #define UART_BAUD_SELECT(baudRate, xtalCpu) ((xtalCpu)/((baudRate)∗16l)-1)

  UART Baudrate Expression.
- #define UART_BAUD_SELECT_DOUBLE_SPEED(baudRate, xtalCpu) (((xtalCpu)/((baudRate)∗8l)-1)|0x8000)

  UART Baudrate Expression for ATmega double speed mode.
- #define UART_RX_BUFFER_SIZE 64
- #define UART_TX_BUFFER_SIZE 64
- #define UART_FRAME_ERROR 0x0800 /∗ Framing Error by UART ∗/
- #define UART_OVERRUN_ERROR 0x0400 /∗ Overrun condition by UART ∗/
- #define UART_BUFFER_OVERFLOW 0x0200 /∗ receive ringbuffer overflow ∗/
- #define UART_NO_DATA 0x0100 /∗ no receive data available ∗/
- #define uart_puts_P(__s) uart_puts_p(PSTR(__s))

  Macro to automatically put a string constant into program memory.
- #define uart1_puts_P(__s) uart1_puts_p(PSTR(__s))

  Macro to automatically put a string constant into program memory.

Functions

- void uart_init (unsigned int baudrate)

  Initialize UART and set baudrate.
- unsigned int uart_getc (void)

  Get received byte from ringbuffer.
- void uart_putc (unsigned char data)

  Put byte to ringbuffer for transmitting via UART.
- void uart_puts (const char ∗s)

  Put string to ringbuffer for transmitting via UART.
- void uart_puts_p (const char ∗progmem_s)

  Put string from program memory to ringbuffer for transmitting via UART.
- void uart1_init (unsigned int baudrate)

  Initialize USART1 (only available on selected ATmegas)

- unsigned int uart1_getc (void)

  Get received byte of USART1 from ringbuffer. (only available on selected ATmega)
- void uart1_putc (unsigned char data)

  Put byte to ringbuffer for transmitting via USART1 (only available on selected ATmega)
- void uart1_puts (const char ∗s)

  Put string to ringbuffer for transmitting via USART1 (only available on selected ATmega)
- void uart1_puts_p (const char ∗s)

  Put string from program memory to ringbuffer for transmitting via USART1 (only available on selected ATmega)

5.1.1 Detailed Description

Interrupt UART library using the built-in UART with transmit and receive circular buffers.

```
#include <uart.h>
```

This library can be used to transmit and receive data through the built in UART.

An interrupt is generated when the UART has finished transmitting or receiving a byte. The interrupt handling routines use circular buffers for buffering received and transmitted data.

The UART_RX_BUFFER_SIZE and UART_TX_BUFFER_SIZE constants define the size of the circular buffers in bytes. Note that these constants must be a power of 2. You may need to adapt this constants to your target and your application by adding CDEFS += -DUART_RX_BUFFER_SIZE=nn -DUART_RX_BUFFER_SIZE=nn to your Makefile.

Note

Based on Atmel Application Note AVR306

Author

Peter Fleury pfleury@gmx.ch http://jump.to/fleury

5.1.2 Macro Definition Documentation

5.1.2.1 #define uart1_puts_P(__s) uart1_puts_p(PSTR(__s))

Macro to automatically put a string constant into program memory.

Definition at line 186 of file uart.h.

5.1.2.2 #define UART_BAUD_SELECT(baudRate, xtalCpu) ((xtalCpu)/((baudRate)∗16l)-1)

UART Baudrate Expression.

constants and macros

Parameters

xtalCpu	system clock in Mhz, e.g. 4000000L for 4Mhz
baudRate	baudrate in bps, e.g. 1200, 2400, 9600

Definition at line 67 of file uart.h.

Referenced by gps_init().

5.1.2.3 #define UART_BAUD_SELECT_DOUBLE_SPEED(baudRate, xtalCpu) (((xtalCpu)/((baudRate)∗8l)-1)|0x8000)

UART Baudrate Expression for ATmega double speed mode.

Parameters

xtalCpu	system clock in Mhz, e.g. 4000000L for 4Mhz
baudRate	baudrate in bps, e.g. 1200, 2400, 9600

Definition at line 73 of file uart.h.

5.1.2.4 #define UART_BUFFER_OVERFLOW 0x0200 /* receive ringbuffer overflow */

Definition at line 95 of file uart.h.

Referenced by gps_get_char(), and SIGNAL().

5.1.2.5 #define UART_FRAME_ERROR 0x0800 /* Framing Error by UART */

test if the size of the circular buffers fits into SRAM

high byte error return code of uart_getc()

Definition at line 93 of file uart.h.

5.1.2.6 #define UART_NO_DATA 0x0100 /* no receive data available */

Definition at line 96 of file uart.h.

Referenced by gps_get_char(), and uart_getc().

5.1.2.7 #define UART_OVERRUN_ERROR 0x0400 /* Overrun condition by UART */

Definition at line 94 of file uart.h.

5.1.2.8 #define uart_puts_P(__s) uart_puts_p(PSTR(__s))

Macro to automatically put a string constant into program memory.

Definition at line 171 of file uart.h.

5.1.2.9 #define UART_RX_BUFFER_SIZE 64

Size of the circular receive buffer, must be power of 2

Definition at line 78 of file uart.h.

5.1.2.10 #define UART_TX_BUFFER_SIZE 64

Size of the circular transmit buffer, must be power of 2

Definition at line 82 of file uart.h.

5.1.3 Function Documentation

5.1.3.1 unsigned int uart1_getc (void)

Get received byte of USART1 from ringbuffer. (only available on selected ATmega)

See also

uart_getc

5.1.3.2 void uart1_init (unsigned int *baudrate*)

Initialize USART1 (only available on selected ATmegas)

See also

uart_init

5.1.3.3 void uart1_putc (unsigned char *data*)

Put byte to ringbuffer for transmitting via USART1 (only available on selected ATmega)

See also

uart_putc

5.1.3.4 void uart1_puts (const char * *s*)

Put string to ringbuffer for transmitting via USART1 (only available on selected ATmega)

See also

uart_puts

5.1.3.5 void uart1_puts_p (const char * *s*)

Put string from program memory to ringbuffer for transmitting via USART1 (only available on selected ATmega)

See also

uart_puts_p

5.1.3.6 unsigned int uart_getc (void)

Get received byte from ringbuffer.

Returns in the lower byte the received character and in the higher byte the last receive error. UART_NO_DATA is returned when no data is available.

Returns

lower byte: received byte from ringbuffer
higher byte: last receive status

- **0** successfully received data from UART
- **UART_NO_DATA**
  no receive data available
- **UART_BUFFER_OVERFLOW**
  Receive ringbuffer overflow. We are not reading the receive buffer fast enough, one or more received character have been dropped

- **UART_OVERRUN_ERROR**
  Overrun condition by UART. A character already present in the UART UDR register was not read by the interrupt handler before the next character arrived, one or more received characters have been dropped.
- **UART_FRAME_ERROR**
  Framing Error by UART

uart_getc() return byte from ringbuffer Returns: lower byte: received byte from ringbuffer higher byte: last receive error < no data available

calculate / store buffer index

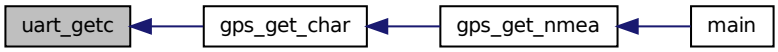
get data from receive buffer

Definition at line 390 of file uart.c.

References UART_NO_DATA, and UART_RX_BUFFER_MASK.

Referenced by gps_get_char().

Here is the caller graph for this function:



5.1.3.7 void uart_init (unsigned int *baudrate*)

Initialize UART and set baudrate.

function prototypes

Parameters

baudrate	Specify baudrate using macro UART_BAUD_SELECT()

uart_init() initialize UART and set baudrate

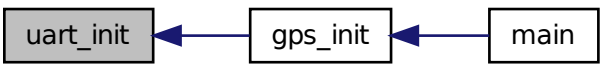
Parameters

in	*baudrate*	using macro UART_BAUD_SELECT()

Definition at line 313 of file uart.c.

Referenced by gps_init().

Here is the caller graph for this function:



5.1.3.8 void uart_putc (unsigned char *data*)

Put byte to ringbuffer for transmitting via UART.

Parameters

data	byte to be transmitted

uart_putc() write byte to ringbuffer for transmitting via UART

Parameters

in	*data*	byte to be transmitted

wait for free space in buffer

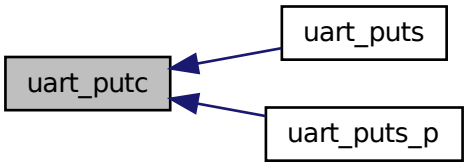
enable UDRE interrupt

Definition at line 416 of file uart.c.

References UART_TX_BUFFER_MASK.

Referenced by uart_puts(), and uart_puts_p().

Here is the caller graph for this function:



5.1.3.9 void uart_puts (const char * *s*)

Put string to ringbuffer for transmitting via UART.

The string is buffered by the uart library in a circular buffer and one character at a time is transmitted to the UART using interrupts. Blocks if it can not write the whole string into the circular buffer.

Parameters

s	string to be transmitted

uart_puts() transmit string to UART

Parameters

in	*s*	string to be transmitted

Definition at line 440 of file uart.c.

References uart_putc().

Here is the call graph for this function:

5.1.3.10 void uart_puts_p (const char * *progmem_s*)

Put string from program memory to ringbuffer for transmitting via UART.

The string is buffered by the uart library in a circular buffer and one character at a time is transmitted to the UART using interrupts. Blocks if it can not write the whole string into the circular buffer.

Parameters

progmem_s	program memory string to be transmitted

See also

uart_puts_P

uart_puts_p() transmit string from program memory to UART

Parameters

in	*progmem_s*	program memory string to be transmitted

Definition at line 452 of file uart.c.

References uart_putc().

Here is the call graph for this function:

Chapter 6

Data Structure Documentation

6.1 afile Struct Reference

`#include <fat16.h>`

Data Fields

- unsigned long start_cluster
- unsigned long cluster_pointer
- unsigned char sector_index
- unsigned int byte_index
- unsigned char mode
- unsigned long filesize
- unsigned long fileposition
- unsigned long sector_in_buffer
- unsigned long directory_sector
- unsigned char directory_index
- unsigned char attribute

6.1.1 Detailed Description

Definition at line 10 of file fat16.h.

6.1.2 Field Documentation

6.1.2.1 unsigned char afile::attribute

Definition at line 22 of file fat16.h.

Referenced by fclose_(), fopen_(), ScanSubDirectories(), and SeekDirectoryEntry().

6.1.2.2 unsigned int afile::byte_index

Definition at line 15 of file fat16.h.

Referenced by fclose_(), fflush_(), fgetchar_(), fopen_(), fputchar_(), and fseek_().

6.1.2.3 unsigned long afile::cluster_pointer

Definition at line 13 of file fat16.h.

Referenced by AppendCluster(), fclose_(), fflush_(), fgetchar_(), fopen_(), fputchar_(), fseek_(), GetFatClusterOffset(), GetNextCluster(), ScanSubDirectories(), and SeekDirectoryEntry().

6.1.2.4 unsigned char afile::directory_index

Definition at line 21 of file fat16.h.

Referenced by CreateDirectoryEntry(), fclose_(), fflush_(), fopen_(), and SeekDirectoryEntry().

6.1.2.5 unsigned long afile::directory_sector

Definition at line 20 of file fat16.h.

Referenced by CreateDirectoryEntry(), fclose_(), fflush_(), fopen_(), and SeekDirectoryEntry().

6.1.2.6 unsigned long afile::fileposition

Definition at line 18 of file fat16.h.

Referenced by fclose_(), fopen_(), fputchar_(), and fseek_().

6.1.2.7 unsigned long afile::filesize

Definition at line 17 of file fat16.h.

Referenced by fclose_(), fflush_(), fgetchar_(), fopen_(), fputchar_(), fseek_(), and SeekDirectoryEntry().

6.1.2.8 unsigned char afile::mode

Definition at line 16 of file fat16.h.

Referenced by fclose_(), fflush_(), and fopen_().

6.1.2.9 unsigned long afile::sector_in_buffer

Definition at line 19 of file fat16.h.

Referenced by fclose_(), fgetchar_(), FindNextFreeCluster(), fopen_(), and GetNextCluster().

6.1.2.10 unsigned char afile::sector_index

Definition at line 14 of file fat16.h.

Referenced by fclose_(), fflush_(), fgetchar_(), fopen_(), fputchar_(), and fseek_().

6.1.2.11 unsigned long afile::start_cluster

Definition at line 12 of file fat16.h.

Referenced by fclose_(), fopen_(), fseek_(), and SeekDirectoryEntry().

The documentation for this struct was generated from the following file:

- fat16.h

6.2 DirEntry Struct Reference

`#include <fat16.h>`

Data Fields

- unsigned char name [8]
- unsigned char extension [3]
- unsigned char attribute
- unsigned char reserved [10]
- unsigned int time
- unsigned int date
- unsigned int startcluster
- unsigned long size

6.2.1 Detailed Description

Definition at line 31 of file fat16.h.

6.2.2 Field Documentation

6.2.2.1 unsigned char DirEntry::attribute

Definition at line 35 of file fat16.h.

Referenced by CreateDirectoryEntry(), and SeekDirectoryEntry().

6.2.2.2 unsigned int DirEntry::date

Definition at line 38 of file fat16.h.

Referenced by fflush_().

6.2.2.3 unsigned char DirEntry::extension[3]

Definition at line 34 of file fat16.h.

6.2.2.4 unsigned char DirEntry::name[8]

Definition at line 33 of file fat16.h.

Referenced by CreateDirectoryEntry(), and SeekDirectoryEntry().

6.2.2.5 unsigned char DirEntry::reserved[10]

Definition at line 36 of file fat16.h.

6.2.2.6 unsigned long DirEntry::size

Definition at line 40 of file fat16.h.

Referenced by CreateDirectoryEntry(), fflush_(), fread_(), fwrite_(), and SeekDirectoryEntry().

6.2.2.7 unsigned int DirEntry::startcluster

Definition at line 39 of file fat16.h.

Referenced by CreateDirectoryEntry(), and SeekDirectoryEntry().

6.2.2.8 unsigned int DirEntry::time

Definition at line 37 of file fat16.h.

Referenced by fflush_().

The documentation for this struct was generated from the following file:

- fat16.h

6.3 FatEntry Struct Reference

`#include <fat16.h>`

Data Fields

- unsigned int next_cluster

6.3.1 Detailed Description

Definition at line 49 of file fat16.h.

6.3.2 Field Documentation

6.3.2.1 unsigned int FatEntry::next_cluster

Definition at line 51 of file fat16.h.

Referenced by AppendCluster(), and FindNextFreeCluster().

The documentation for this struct was generated from the following file:

- fat16.h

Chapter 7

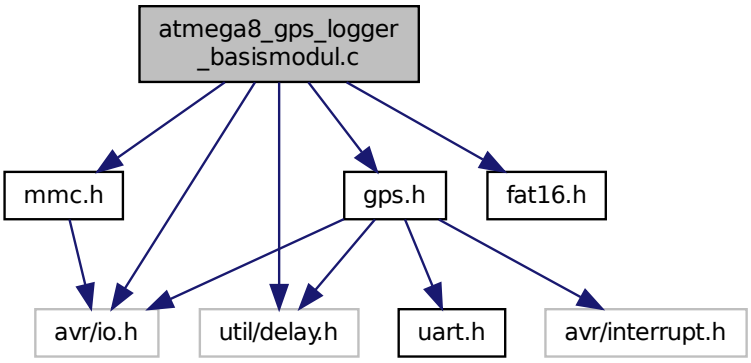
File Documentation

7.1 atmega8_gps_logger_basismodul.c File Reference

Hauptfile des Projekts ATmega8 GPS Logger.

```
#include <avr/io.h>
#include <util/delay.h>
#include "gps.h"
#include "mmc.h"
#include "fat16.h"
```

Include dependency graph for atmega8_gps_logger_basismodul.c:



Macros

- #define F_CPU 16000000L
- #define LED_PORT PORTD
- #define LED_STAT PD6
- #define LED_WARN PD7
- #define LEDCODE_OFF LED_PORT &= ~((1 << LED_WARN) | (1 << LED_STAT))
- #define LEDCODE_OK LED_PORT = (LED_PORT & ~(1 << LED_WARN)) | (1 << LED_STAT)
- #define LEDCODE_WARNING LED_PORT = (LED_PORT & ~(1 << LED_STAT)) | (1 << LED_WARN)
- #define LEDCODE_PROCESSING LED_PORT |= (1 << LED_WARN) | (1 << LED_STAT)
- #define LEDCODE_BLINK LED_PORT ^= (1 << LED_STAT)

Functions

- int main (void)

  Hauptfunktion.

Variables

- struct {
  unsigned record:1
  unsigned keydown:1
  } status

7.1.1 Detailed Description

Hauptfile des Projekts ATmega8 GPS Logger.

Author

Martin Matysiak (mail@k621.de)
V. Pippan (webmaster@vpippan.at)

Date

2013-10-18

Version

20131018

Todo

- Code schöner schreiben, Kommtentare einfügen, Doxygen Doku verbessern
- FAT, UART und MMC Libraries in eigenem Ordner speichern
- In neuer Version auf FAT verzichten und Daten ohne Dateisystem auf SD Karte schreiben (siehe g↩ LoggerMini) -> weniger Speicher für Code, schneller?
- Wartezeit für GPS Datenempfang einstellbar machen (eventuell nur Positionsdaten auslesen?)
- GPS Maus mit anderen Einstellungen initialisieren (höhere Datenrate, andere NMEA Sätze?)

- Start / Stop aus dem File entfernen -> Überprüfen ob Datei vorhanden und dann neue anlegen (Dateinamen bei jedem starten ändern und damit neue Datei anlegen -> String append ?)
- Akkuspannung messen und bei zu niedriger Spannung -> sleep mode (Datei abschließen), GPS und Zusatzmodul abschalten und warnen
- GPS nur einschalten wenn Aufzeichnung aktiviert, sonst ausschalten
- GPS Daten auf Display ausgeben, zusätzliche Sensoren, Jeti Kommunikation -> größerer uC benötigt? -> ATmega32?
- Berechnungen für Oldtimerrallye Training machen und am Display ausgeben
- Erkennung von Zusatzmodulen und Code für diese
- Bei Verwendung mit LCD Zusatzmodul das Logging abschaltbar machen -> nur Anzeige der Daten

Definition in file atmega8_gps_logger_basismodul.c.

7.1.2 Macro Definition Documentation

7.1.2.1 #define F_CPU 16000000L

Definition at line 51 of file atmega8_gps_logger_basismodul.c.

Referenced by gps_init().

7.1.2.2 #define LED_PORT PORTD

Definition at line 53 of file atmega8_gps_logger_basismodul.c.

7.1.2.3 #define LED_STAT PD6

Definition at line 54 of file atmega8_gps_logger_basismodul.c.

7.1.2.4 #define LED_WARN PD7

Definition at line 55 of file atmega8_gps_logger_basismodul.c.

7.1.2.5 #define LEDCODE_BLINK LED_PORT ^= (1 << LED_STAT)

Definition at line 61 of file atmega8_gps_logger_basismodul.c.

Referenced by main().

7.1.2.6 #define LEDCODE_OFF LED_PORT &= ~((1 << LED_WARN) | (1 << LED_STAT))

Definition at line 57 of file atmega8_gps_logger_basismodul.c.

Referenced by main().

7.1.2.7 #define LEDCODE_OK LED_PORT = (LED_PORT & ~(1 << LED_WARN)) | (1 << LED_STAT)

Definition at line 58 of file atmega8_gps_logger_basismodul.c.

Referenced by main().

7.1.2.8 #define LEDCODE_PROCESSING LED_PORT |= (1 << LED_WARN) | (1 << LED_STAT)

Definition at line 60 of file atmega8_gps_logger_basismodul.c.

Referenced by main().

7.1.2.9 #define LEDCODE_WARNING LED_PORT = (LED_PORT & ~(1 << LED_STAT)) | (1 << LED_WARN)

Definition at line 59 of file atmega8_gps_logger_basismodul.c.

Referenced by main().

7.1.3 Function Documentation

7.1.3.1 int main (void)

Hauptfunktion.

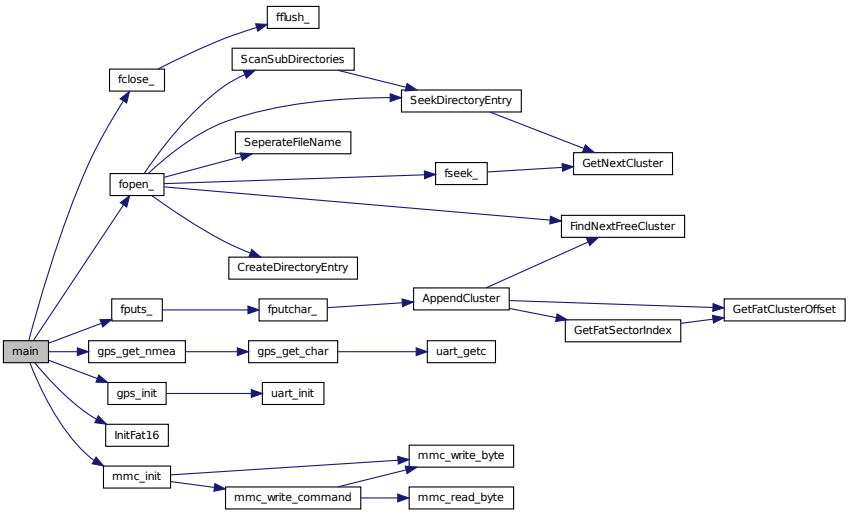
Die Hauptfunktion enthält die Initialisierungen beim Programmstart und die Hauptschleife des Programms. < definiert ein character feld für nmea daten mit 128 zeichen länge

Hauptschleife

Definition at line 74 of file atmega8_gps_logger_basismodul.c.

References fclose_(), fopen_(), fputs_(), gps_get_nmea(), gps_init(), InitFat16(), LEDCODE_BLINK, LEDCODE_↩ OFF, LEDCODE_OK, LEDCODE_PROCESSING, LEDCODE_WARNING, mmc_init(), and status.

Here is the call graph for this function:



7.1.4 Variable Documentation

7.1.4.1 unsigned keydown

Definition at line 66 of file atmega8_gps_logger_basismodul.c.

7.1.4.2 unsigned record

Definition at line 65 of file atmega8_gps_logger_basismodul.c.

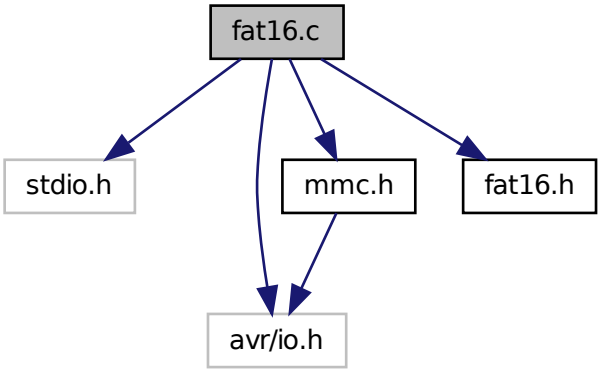
7.1.4.3 struct { ... } status

Referenced by main().

7.2 fat16.c File Reference

```
#include <stdio.h>
#include <avr/io.h>
#include "mmc.h"
#include "fat16.h"
```

Include dependency graph for fat16.c:



Functions

- unsigned char InitFat16 (void)
- unsigned char fopen_ (unsigned char *fname, char mode, File *file)
- int fflush_ (File *file)
- void fclose_ (File *file)
- unsigned long fread_ (void *buffer, unsigned long size, unsigned long count, File *file)
- unsigned long fwrite_ (void *buffer, unsigned long size, unsigned long count, File *file)
- int fseek_ (File *file, long offset, int origin)
- int fgetchar_ (File *file)
- unsigned char fputchar_ (File *file, char c)
- unsigned char fputs_ (File *file, char *string)
- char * fgets_ (char *string, int count, File *file)
- unsigned char fexist_ (unsigned char *fname, File *file)
- unsigned int GetNextCluster (File *file)
- unsigned int FindNextFreeCluster (File *file)
- unsigned char AppendCluster (File *file)
- unsigned int GetFatClusterOffset (File *file)

- unsigned int GetFatSectorIndex (File *file)
- unsigned char CreateDirectoryEntry (unsigned char *fname, unsigned int cluster, File *file, unsigned char attrib)
- unsigned char SeekDirectoryEntry (unsigned char *fname, File *file)
- unsigned char ScanSubDirectories (unsigned char *fname, File *file)
- void SeperateFileName (unsigned char *fname, unsigned char *name)

Variables

- unsigned char SectorsPerCluster = 0
- unsigned char FatCopies = 0
- unsigned int PossibleRootEntries = 0
- unsigned int SectorsPerFat = 0
- unsigned long ReservedSectors = 0
- unsigned long FirstPartitionSector = 0
- unsigned long FileAllocationTable = 0
- unsigned long RootDirectory = 0
- unsigned long FirstDataCluster = 0
- unsigned char FileBuffer [512]
- struct DirEntry * DirectoryEntry
- struct FatEntry * Fat

7.2.1 Function Documentation

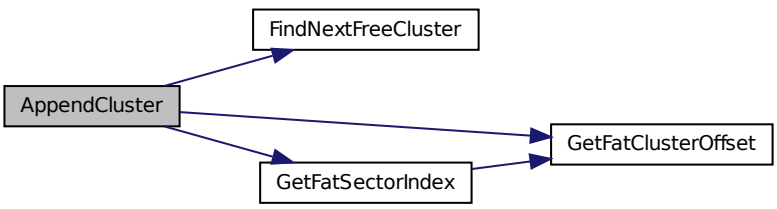
7.2.1.1 unsigned char AppendCluster (File * *file*)

Definition at line 614 of file fat16.c.

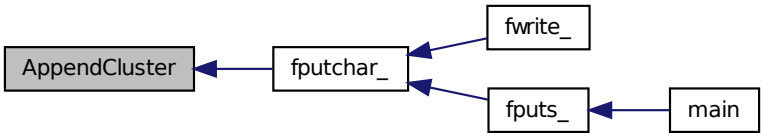
References afile::cluster_pointer, FileAllocationTable, FileBuffer, FindNextFreeCluster(), FirstDataCluster, GetFatClusterOffset(), GetFatSectorIndex(), FatEntry::next_cluster, SDC_GetSector, SDC_PutSector, and SectorsPerCluster.

Referenced by fputchar_().

Here is the call graph for this function:



Here is the caller graph for this function:



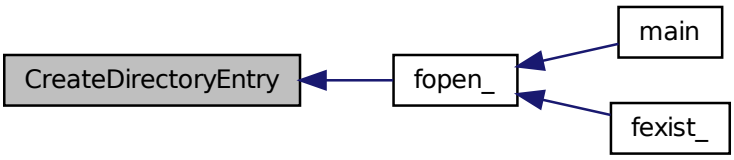
7.2.1.2 unsigned char CreateDirectoryEntry (unsigned char * *fname,* unsigned int *cluster,* File * *file,* unsigned char *attrib*)

Definition at line 687 of file fat16.c.

References DirEntry::attribute, afile::directory_index, afile::directory_sector, FileBuffer, DirEntry::name, PossibleRootEntries, RootDirectory, SDC_GetSector, SDC_PutSector, DirEntry::size, and DirEntry::startcluster.

Referenced by fopen_().

Here is the caller graph for this function:



7.2.1.3 void fclose_ (File * *file*)

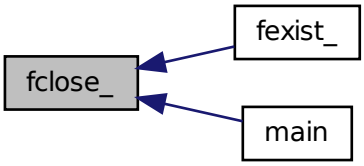
Definition at line 195 of file fat16.c.

References afile::attribute, afile::byte_index, afile::cluster_pointer, afile::directory_index, afile::directory_sector, fflush_(), afile::fileposition, afile::filesize, afile::mode, afile::sector_in_buffer, afile::sector_index, and afile::start_cluster.

Referenced by fexist_(), and main().

Here is the call graph for this function:

Here is the caller graph for this function:

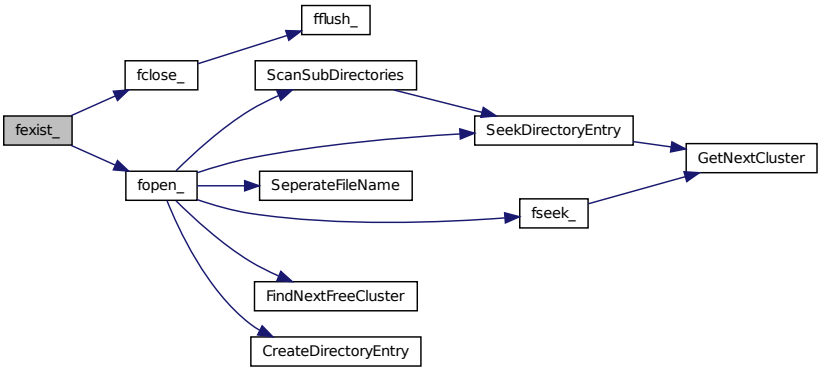


7.2.1.4 unsigned char fexist_ (unsigned char * *fname,* File * *file*)

Definition at line 505 of file fat16.c.

References fclose_(), and fopen_().

Here is the call graph for this function:



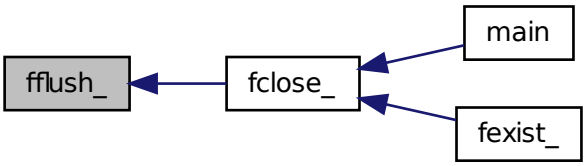
7.2.1.5 int fflush_ (File * *file*)

Definition at line 159 of file fat16.c.

References afile::byte_index, afile::cluster_pointer, DirEntry::date, afile::directory_index, afile::directory_sector, FileBuffer, afile::filesize, afile::mode, SDC_GetSector, SDC_PutSector, afile::sector_index, DirEntry::size, and Dir↩ Entry::time.

Referenced by fclose_().

Here is the caller graph for this function:



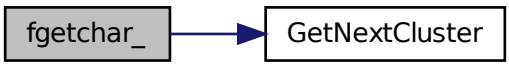
7.2.1.6 int fgetchar_ (File * *file*)

Definition at line 356 of file fat16.c.

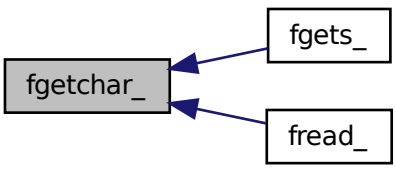
References afile::byte_index, afile::cluster_pointer, FileBuffer, afile::filesize, GetNextCluster(), SDC_GetSector, afile::sector_in_buffer, afile::sector_index, and SectorsPerCluster.

Referenced by fgets_(), and fread_().

Here is the call graph for this function:



Here is the caller graph for this function:

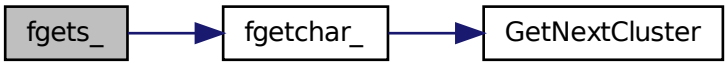


7.2.1.7 char* fgets_ (char * *string,* int *count,* File * *file*)

Definition at line 463 of file fat16.c.

References fgetchar_().

Here is the call graph for this function:



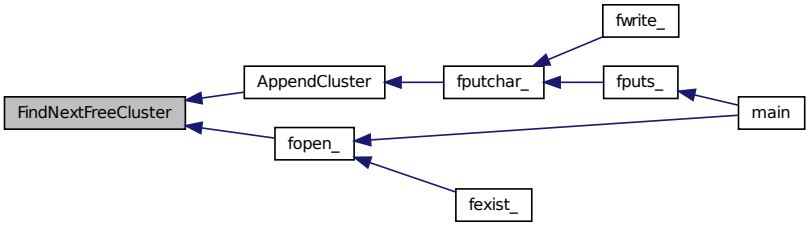
7.2.1.8 unsigned int FindNextFreeCluster (File * *file*)

Definition at line 574 of file fat16.c.

References FileAllocationTable, FileBuffer, FatEntry::next_cluster, SDC_GetSector, SDC_PutSector, afile::sector_in_buffer, and SectorsPerFat.

Referenced by AppendCluster(), and fopen_().

Here is the caller graph for this function:



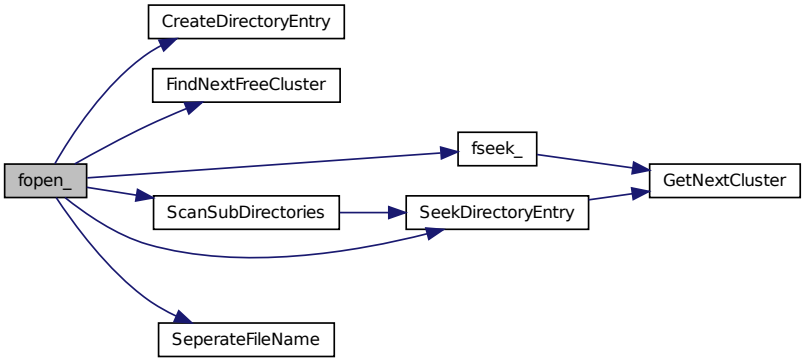
7.2.1.9 unsigned char fopen_ (unsigned char * *fname,* char *mode,* File * *file*)

Definition at line 97 of file fat16.c.

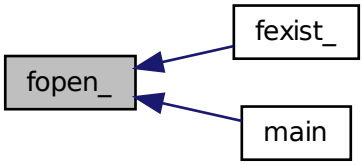
References _FILE, afile::attribute, afile::byte_index, afile::cluster_pointer, CreateDirectoryEntry(), afile::directory_index, afile::directory_sector, afile::fileposition, afile::filesize, FindNextFreeCluster(), FirstDataCluster, fseek_(), afile::mode, ScanSubDirectories(), afile::sector_in_buffer, afile::sector_index, SectorsPerCluster, SeekDirectoryEntry(), SeperateFileName(), and afile::start_cluster.

Referenced by fexist_(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



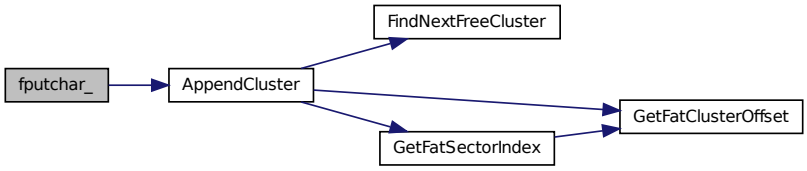
7.2.1.10 unsigned char fputchar_ (File * *file,* char *c*)

Definition at line 400 of file fat16.c.

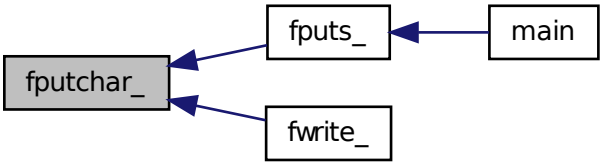
References AppendCluster(), afile::byte_index, afile::cluster_pointer, FileBuffer, afile::fileposition, afile::filesize, SDC_PutSector, afile::sector_index, and SectorsPerCluster.

Referenced by fputs_(), and fwrite_().

Here is the call graph for this function:



Here is the caller graph for this function:



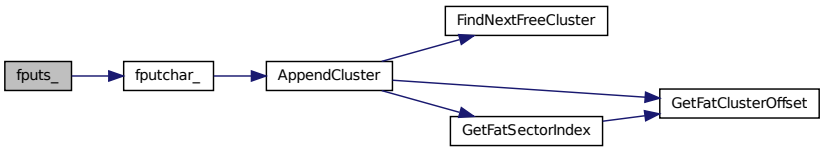
7.2.1.11 unsigned char fputs_ (File * *file,* char * *string*)

Definition at line 443 of file fat16.c.

References fputchar_().

Referenced by main().

Here is the call graph for this function:



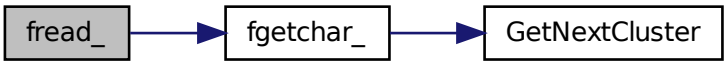
Here is the caller graph for this function:

7.2.1.12 unsigned long fread_ (void * *buffer,* unsigned long *size,* unsigned long *count,* File * *file*)

Definition at line 221 of file fat16.c.

References fgetchar_(), and DirEntry::size.

Here is the call graph for this function:



7.2.1.13 int fseek_ (File * *file,* long *offset,* int *origin*)

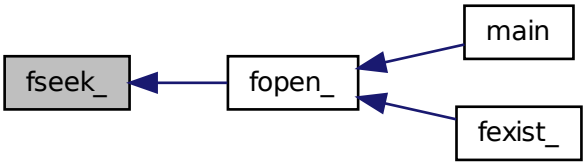
Definition at line 286 of file fat16.c.

References afile::byte_index, afile::cluster_pointer, FileBuffer, afile::fileposition, afile::filesize, GetNextCluster(), SDC_GetSector, afile::sector_index, SectorsPerCluster, and afile::start_cluster.

Referenced by fopen_().

Here is the call graph for this function:

Here is the caller graph for this function:

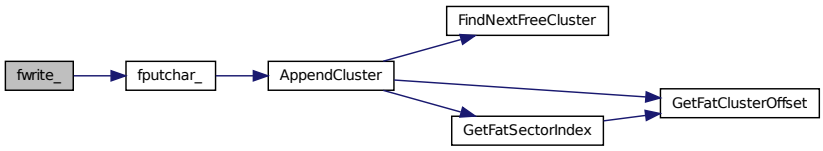


7.2.1.14 unsigned long fwrite_ (void * *buffer,* unsigned long *size,* unsigned long *count,* File * *file*)

Definition at line 254 of file fat16.c.

References fputchar_(), and DirEntry::size.

Here is the call graph for this function:



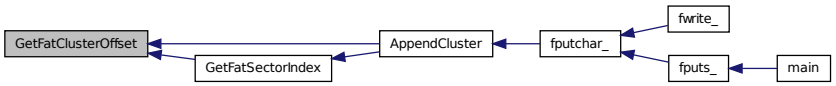
7.2.1.15 unsigned int GetFatClusterOffset (File * *file*)

Definition at line 646 of file fat16.c.

References afile::cluster_pointer, FirstDataCluster, and SectorsPerCluster.

Referenced by AppendCluster(), and GetFatSectorIndex().

Here is the caller graph for this function:



7.2.1.16 unsigned int GetFatSectorIndex (File * *file*)

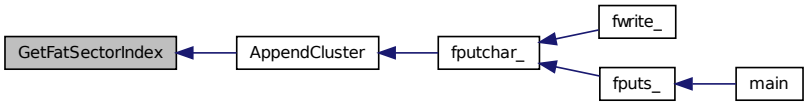
Definition at line 664 of file fat16.c.

References GetFatClusterOffset().

Referenced by AppendCluster().

Here is the call graph for this function:

Here is the caller graph for this function:



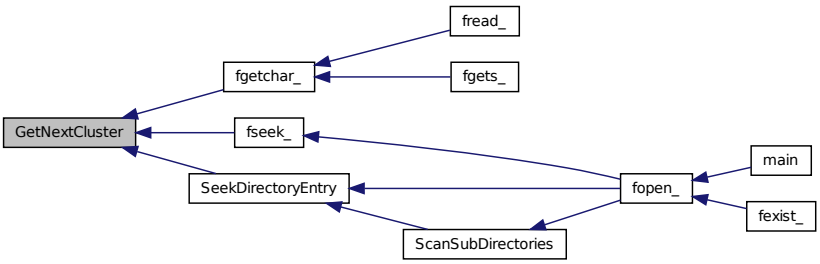
7.2.1.17 unsigned int GetNextCluster (File * *file*)

Definition at line 526 of file fat16.c.

References afile::cluster_pointer, FileAllocationTable, FileBuffer, FirstDataCluster, RootDirectory, SDC_GetSector, afile::sector_in_buffer, and SectorsPerCluster.

Referenced by fgetchar_(), fseek_(), and SeekDirectoryEntry().

Here is the caller graph for this function:



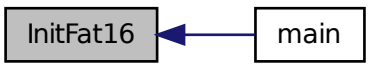
7.2.1.18 unsigned char InitFat16 (void)

Definition at line 71 of file fat16.c.

References FatCopies, FileAllocationTable, FileBuffer, FirstDataCluster, FirstPartitionSector, MBR_SECTOR, PossibleRootEntries, ReservedSectors, RootDirectory, SDC_GetSector, SectorsPerCluster, and SectorsPerFat.

Referenced by main().

Here is the caller graph for this function:



7.2.1.19 unsigned char ScanSubDirectories (unsigned char * *fname,* File * *file*)

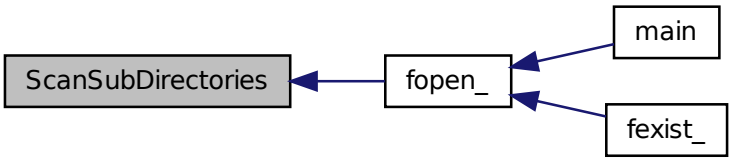
Definition at line 790 of file fat16.c.

References _DIRECTORY, _FILE, afile::attribute, afile::cluster_pointer, RootDirectory, and SeekDirectoryEntry().

Referenced by fopen_().

Here is the call graph for this function:

Here is the caller graph for this function:



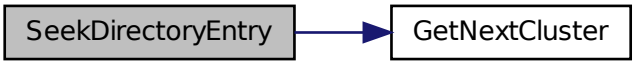
7.2.1.20 unsigned char SeekDirectoryEntry (unsigned char * *fname,* File * *file*)

Definition at line 736 of file fat16.c.

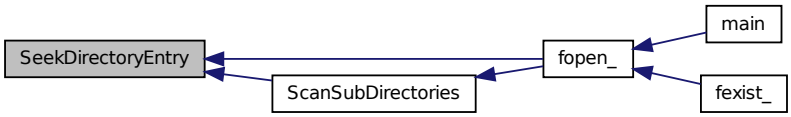
References afile::attribute, DirEntry::attribute, afile::cluster_pointer, afile::directory_index, afile::directory_sector, FileBuffer, afile::filesize, FirstDataCluster, GetNextCluster(), DirEntry::name, SDC_GetSector, SectorsPerCluster, DirEntry::size, afile::start_cluster, and DirEntry::startcluster.

Referenced by fopen_(), and ScanSubDirectories().

Here is the call graph for this function:



Here is the caller graph for this function:

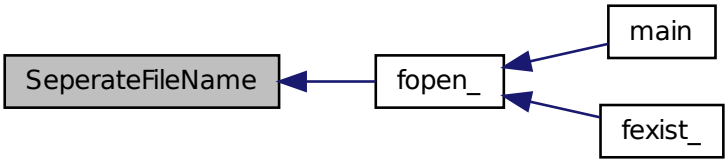


7.2.1.21 void SeperateFileName (unsigned char * *fname,* unsigned char * *name*)

Definition at line 854 of file fat16.c.

Referenced by fopen_().

Here is the caller graph for this function:



7.2.2 Variable Documentation

7.2.2.1 struct DirEntry* DirectoryEntry

Definition at line 56 of file fat16.c.

7.2.2.2 struct FatEntry* Fat

Definition at line 57 of file fat16.c.

7.2.2.3 unsigned char FatCopies = 0

Definition at line 46 of file fat16.c.

Referenced by InitFat16().

7.2.2.4 unsigned long FileAllocationTable = 0

Definition at line 51 of file fat16.c.

Referenced by AppendCluster(), FindNextFreeCluster(), GetNextCluster(), and InitFat16().

7.2.2.5 unsigned char FileBuffer[512]

Definition at line 55 of file fat16.c.

Referenced by AppendCluster(), CreateDirectoryEntry(), fflush_(), fgetchar_(), FindNextFreeCluster(), fputchar_(), fseek_(), GetNextCluster(), InitFat16(), and SeekDirectoryEntry().

7.2.2.6 unsigned long FirstDataCluster = 0

Definition at line 53 of file fat16.c.

Referenced by AppendCluster(), fopen_(), GetFatClusterOffset(), GetNextCluster(), InitFat16(), and Seek↩ DirectoryEntry().

7.2.2.7 unsigned long FirstPartitionSector = 0

Definition at line 50 of file fat16.c.

Referenced by InitFat16().

7.2.2.8 unsigned int PossibleRootEntries = 0

Definition at line 47 of file fat16.c.

Referenced by CreateDirectoryEntry(), and InitFat16().

7.2.2.9 unsigned long ReservedSectors = 0

Definition at line 49 of file fat16.c.

Referenced by InitFat16().

7.2.2.10 unsigned long RootDirectory = 0

Definition at line 52 of file fat16.c.

Referenced by CreateDirectoryEntry(), GetNextCluster(), InitFat16(), and ScanSubDirectories().

7.2.2.11 unsigned char SectorsPerCluster = 0

Definition at line 45 of file fat16.c.

Referenced by AppendCluster(), fgetchar_(), fopen_(), fputchar_(), fseek_(), GetFatClusterOffset(), GetNext↩ Cluster(), InitFat16(), and SeekDirectoryEntry().

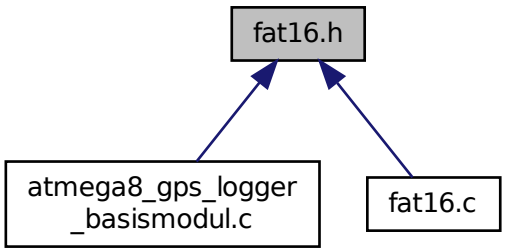
7.2.2.12 unsigned int SectorsPerFat = 0

Definition at line 48 of file fat16.c.

Referenced by FindNextFreeCluster(), and InitFat16().

7.3 fat16.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct afile
- struct DirEntry
- struct FatEntry

Macros

- #define MBR_SECTOR 0
- #define _UNUSED 1
- #define _ARCHIVE 2
- #define _READ_ONLY 4
- #define _SYSTEM 8
- #define _DIRECTORY 16
- #define _FILE 32

Typedefs

- typedef struct afile File

Functions

- unsigned char InitFat16 (void)
- unsigned char fopen_ (unsigned char ∗fname, char mode, File ∗file)
- int fflush_ (File ∗file)
- void fclose_ (File ∗file)
- unsigned long fread_ (void ∗buffer, unsigned long size, unsigned long count, File ∗file)
- unsigned long fwrite_ (void ∗buffer, unsigned long size, unsigned long count, File ∗file)
- int fseek_ (File ∗file, long offset, int origin)
- int fgetchar_ (File ∗file)
- unsigned char fputchar_ (File ∗file, char c)
- unsigned char fputs_ (File ∗file, char ∗string)
- char ∗ fgets_ (char ∗s, int count, File ∗file)

- int rename (char *oldname, char *newname)
- unsigned char fexist_ (unsigned char *fname, File *file)
- unsigned char CreateDirectoryEntry (unsigned char *fname, unsigned int cluster, File *file, unsigned char attrib)
- unsigned int FindNextFreeCluster (File *file)
- unsigned char SeekDirectoryEntry (unsigned char *fname, File *file)
- void SeperateFileName (unsigned char *fname, unsigned char *name)
- unsigned char ScanSubDirectories (unsigned char *fname, File *file)
- unsigned int GetNextCluster (File *file)
- unsigned char AppendCluster (File *file)
- unsigned int GetFatClusterOffset (File *file)
- unsigned int GetFatSectorIndex (File *file)

Variables

- unsigned char SectorsPerCluster
- unsigned char FileBuffer [512]
- File myfile

7.3.1 Macro Definition Documentation

7.3.1.1 #define _ARCHIVE 2

Definition at line 99 of file fat16.h.

7.3.1.2 #define _DIRECTORY 16

Definition at line 102 of file fat16.h.

Referenced by ScanSubDirectories().

7.3.1.3 #define _FILE 32

Definition at line 103 of file fat16.h.

Referenced by fopen_(), and ScanSubDirectories().

7.3.1.4 #define _READ_ONLY 4

Definition at line 100 of file fat16.h.

7.3.1.5 #define _SYSTEM 8

Definition at line 101 of file fat16.h.

7.3.1.6 #define _UNUSED 1

Definition at line 98 of file fat16.h.

7.3.1.7 #define MBR_SECTOR 0

Definition at line 97 of file fat16.h.

Referenced by InitFat16().

7.3.2 Typedef Documentation

7.3.2.1 typedef struct afile File

7.3.3 Function Documentation

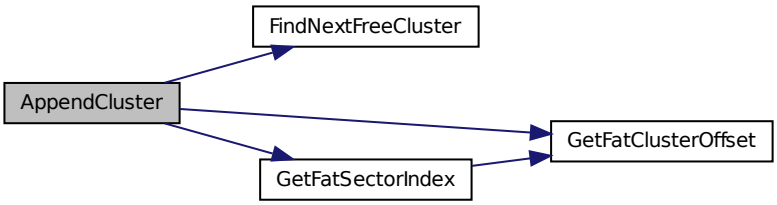
7.3.3.1 unsigned char AppendCluster (File * *file*)

Definition at line 614 of file fat16.c.

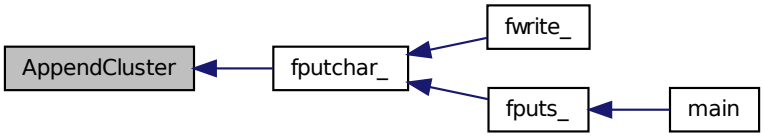
References afile::cluster_pointer, FileAllocationTable, FileBuffer, FindNextFreeCluster(), FirstDataCluster, GetFatClusterOffset(), GetFatSectorIndex(), FatEntry::next_cluster, SDC_GetSector, SDC_PutSector, and SectorsPerCluster.

Referenced by fputchar_().

Here is the call graph for this function:



Here is the caller graph for this function:



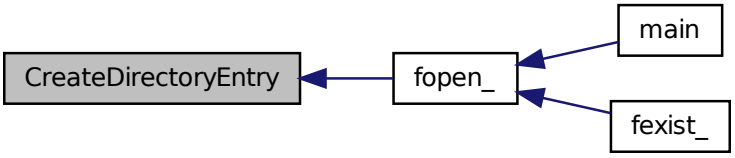
7.3.3.2 unsigned char CreateDirectoryEntry (unsigned char * *fname,* unsigned int *cluster,* File * *file,* unsigned char *attrib*)

Definition at line 687 of file fat16.c.

References DirEntry::attribute, afile::directory_index, afile::directory_sector, FileBuffer, DirEntry::name, PossibleRootEntries, RootDirectory, SDC_GetSector, SDC_PutSector, DirEntry::size, and DirEntry::startcluster.

Referenced by fopen_().

Here is the caller graph for this function:



7.3.3.3 void fclose_ (File * *file*)

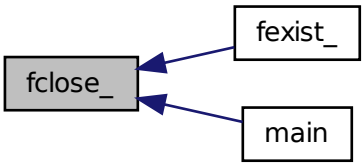
Definition at line 195 of file fat16.c.

References afile::attribute, afile::byte_index, afile::cluster_pointer, afile::directory_index, afile::directory_sector, fflush_(), afile::fileposition, afile::filesize, afile::mode, afile::sector_in_buffer, afile::sector_index, and afile::start_cluster.

Referenced by fexist_(), and main().

Here is the call graph for this function:

Here is the caller graph for this function:

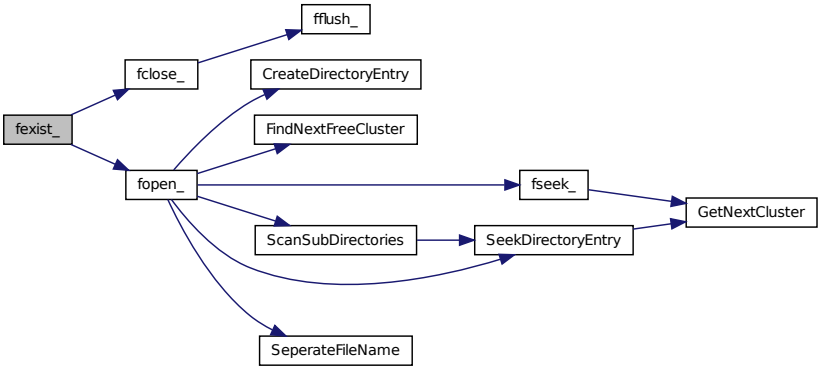


7.3.3.4 unsigned char fexist_ (unsigned char * *fname,* File * *file*)

Definition at line 505 of file fat16.c.

References fclose_(), and fopen_().

Here is the call graph for this function:



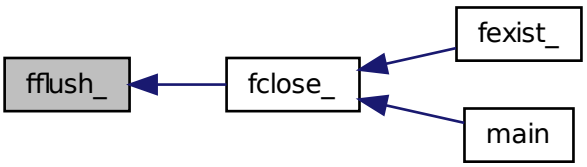
7.3.3.5 int fflush_ (File * *file*)

Definition at line 159 of file fat16.c.

References afile::byte_index, afile::cluster_pointer, DirEntry::date, afile::directory_index, afile::directory_sector, FileBuffer, afile::filesize, afile::mode, SDC_GetSector, SDC_PutSector, afile::sector_index, DirEntry::size, and Dir↩ Entry::time.

Referenced by fclose_().

Here is the caller graph for this function:



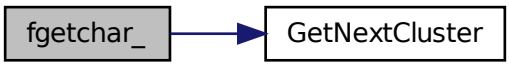
7.3.3.6 int fgetchar_ (File * *file*)

Definition at line 356 of file fat16.c.

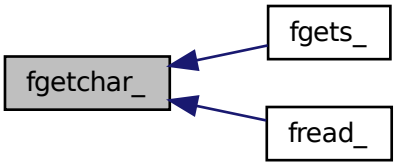
References afile::byte_index, afile::cluster_pointer, FileBuffer, afile::filesize, GetNextCluster(), SDC_GetSector, afile::sector_in_buffer, afile::sector_index, and SectorsPerCluster.

Referenced by fgets_(), and fread_().

Here is the call graph for this function:



Here is the caller graph for this function:

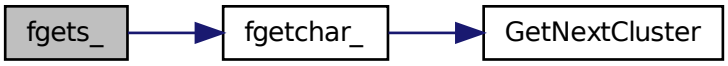


7.3.3.7 char* fgets_ (char * *s*, int *count*, File * *file*)

Definition at line 463 of file fat16.c.

References fgetchar_().

Here is the call graph for this function:



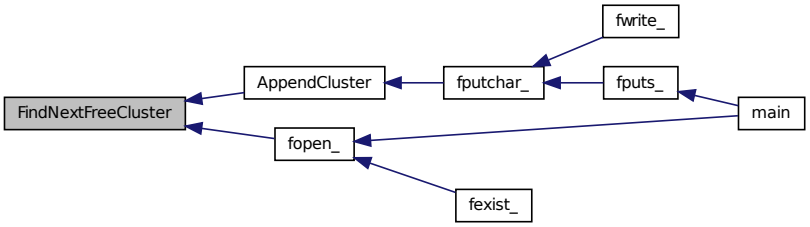
7.3.3.8 unsigned int FindNextFreeCluster (File * *file*)

Definition at line 574 of file fat16.c.

References FileAllocationTable, FileBuffer, FatEntry::next_cluster, SDC_GetSector, SDC_PutSector, afile::sector↩
_in_buffer, and SectorsPerFat.

Referenced by AppendCluster(), and fopen_().

Here is the caller graph for this function:



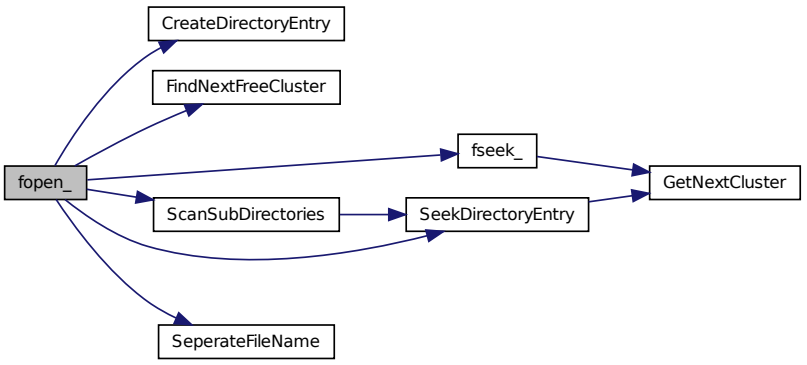
7.3.3.9 unsigned char fopen_ (unsigned char * *fname*, char *mode*, File * *file*)

Definition at line 97 of file fat16.c.

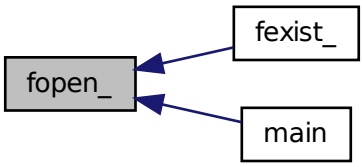
References _FILE, afile::attribute, afile::byte_index, afile::cluster_pointer, CreateDirectoryEntry(), afile::directory_index, afile::directory_sector, afile::fileposition, afile::filesize, FindNextFreeCluster(), FirstDataCluster, fseek_(), afile::mode, ScanSubDirectories(), afile::sector_in_buffer, afile::sector_index, SectorsPerCluster, SeekDirectoryEntry(), SeperateFileName(), and afile::start_cluster.

Referenced by fexist_(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:



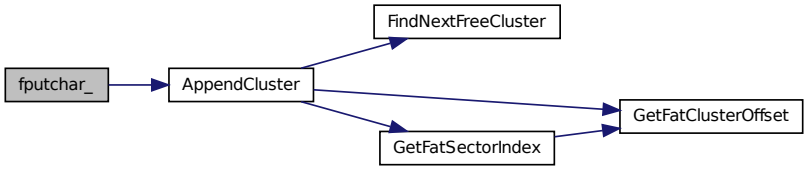
7.3.3.10 unsigned char fputchar_ (File * *file*, char *c*)

Definition at line 400 of file fat16.c.

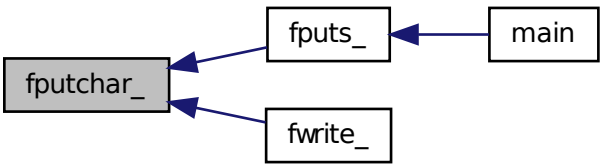
References AppendCluster(), afile::byte_index, afile::cluster_pointer, FileBuffer, afile::fileposition, afile::filesize, SDC_PutSector, afile::sector_index, and SectorsPerCluster.

Referenced by fputs_(), and fwrite_().

Here is the call graph for this function:



Here is the caller graph for this function:



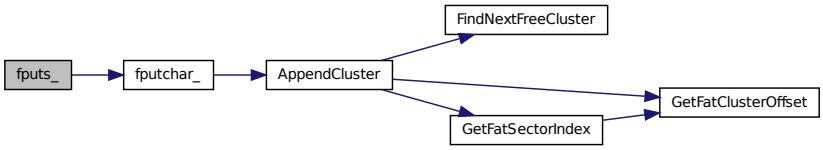
7.3.3.11 unsigned char fputs_ (File * *file*, char * *string*)

Definition at line 443 of file fat16.c.

References fputchar_().

Referenced by main().

Here is the call graph for this function:



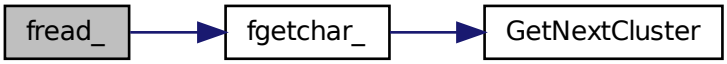
Here is the caller graph for this function:

7.3.3.12 unsigned long fread_ (void * *buffer*, unsigned long *size*, unsigned long *count*, File * *file*)

Definition at line 221 of file fat16.c.

References fgetchar_(), and DirEntry::size.

Here is the call graph for this function:



7.3.3.13 int fseek_ (File * *file*, long *offset*, int *origin*)

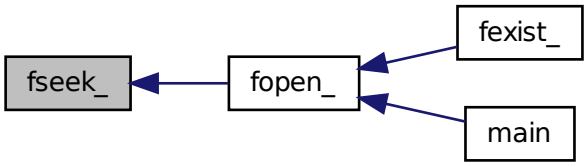
Definition at line 286 of file fat16.c.

References afile::byte_index, afile::cluster_pointer, FileBuffer, afile::fileposition, afile::filesize, GetNextCluster(), SDC_GetSector, afile::sector_index, SectorsPerCluster, and afile::start_cluster.

Referenced by fopen_().

Here is the call graph for this function:

Here is the caller graph for this function:

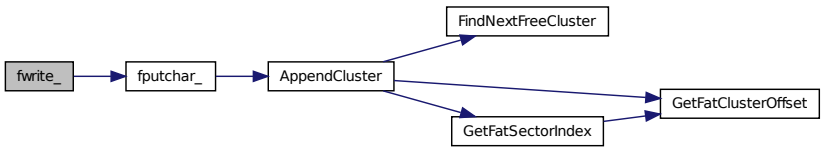


7.3.3.14 unsigned long fwrite_ (void * *buffer,* unsigned long *size,* unsigned long *count,* File * *file*)

Definition at line 254 of file fat16.c.

References fputchar_(), and DirEntry::size.

Here is the call graph for this function:



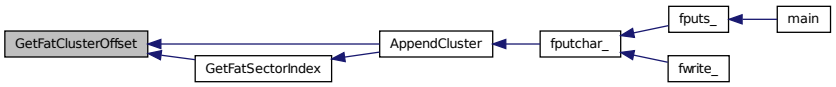
7.3.3.15 unsigned int GetFatClusterOffset (File * *file*)

Definition at line 646 of file fat16.c.

References afile::cluster_pointer, FirstDataCluster, and SectorsPerCluster.

Referenced by AppendCluster(), and GetFatSectorIndex().

Here is the caller graph for this function:



7.3.3.16 unsigned int GetFatSectorIndex (File * *file*)

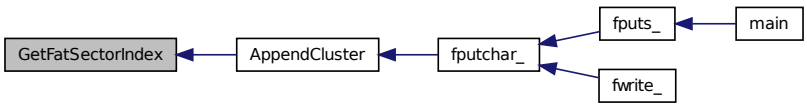
Definition at line 664 of file fat16.c.

References GetFatClusterOffset().

Referenced by AppendCluster().

Here is the call graph for this function:

Here is the caller graph for this function:



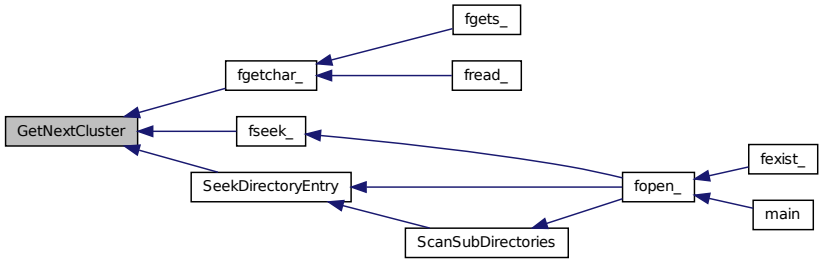
7.3.3.17 unsigned int GetNextCluster (File * *file*)

Definition at line 526 of file fat16.c.

References afile::cluster_pointer, FileAllocationTable, FileBuffer, FirstDataCluster, RootDirectory, SDC_GetSector, afile::sector_in_buffer, and SectorsPerCluster.

Referenced by fgetchar_(), fseek_(), and SeekDirectoryEntry().

Here is the caller graph for this function:



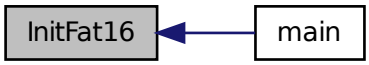
7.3.3.18 unsigned char InitFat16 (void)

Definition at line 71 of file fat16.c.

References FatCopies, FileAllocationTable, FileBuffer, FirstDataCluster, FirstPartitionSector, MBR_SECTOR, PossibleRootEntries, ReservedSectors, RootDirectory, SDC_GetSector, SectorsPerCluster, and SectorsPerFat.

Referenced by main().

Here is the caller graph for this function:



7.3.3.19 int rename (char * *oldname,* char * *newname*)

7.3.3.20 unsigned char ScanSubDirectories (unsigned char * *fname,* File * *file*)

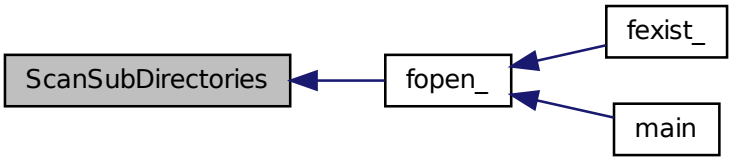
Definition at line 790 of file fat16.c.

References _DIRECTORY, _FILE, afile::attribute, afile::cluster_pointer, RootDirectory, and SeekDirectoryEntry().

Referenced by fopen_().

Here is the call graph for this function:

Here is the caller graph for this function:



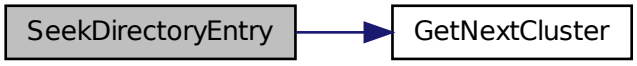
7.3.3.21 unsigned char SeekDirectoryEntry (unsigned char * *fname,* File * *file*)

Definition at line 736 of file fat16.c.

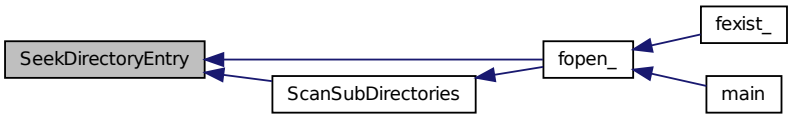
References afile::attribute, DirEntry::attribute, afile::cluster_pointer, afile::directory_index, afile::directory_sector, FileBuffer, afile::filesize, FirstDataCluster, GetNextCluster(), DirEntry::name, SDC_GetSector, SectorsPerCluster, DirEntry::size, afile::start_cluster, and DirEntry::startcluster.

Referenced by fopen_(), and ScanSubDirectories().

Here is the call graph for this function:



Here is the caller graph for this function:

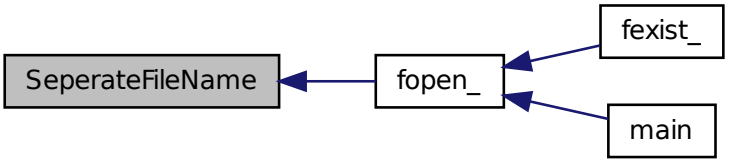


7.3.3.22 void SeperateFileName (unsigned char * *fname,* unsigned char * *name*)

Definition at line 854 of file fat16.c.

Referenced by fopen_().

Here is the caller graph for this function:



7.3.4 Variable Documentation

7.3.4.1 unsigned char FileBuffer[512]

Definition at line 55 of file fat16.c.

Referenced by AppendCluster(), CreateDirectoryEntry(), fflush_(), fgetchar_(), FindNextFreeCluster(), fputchar_(), fseek_(), GetNextCluster(), InitFat16(), and SeekDirectoryEntry().

7.3.4.2 File myfile

7.3.4.3 unsigned char SectorsPerCluster

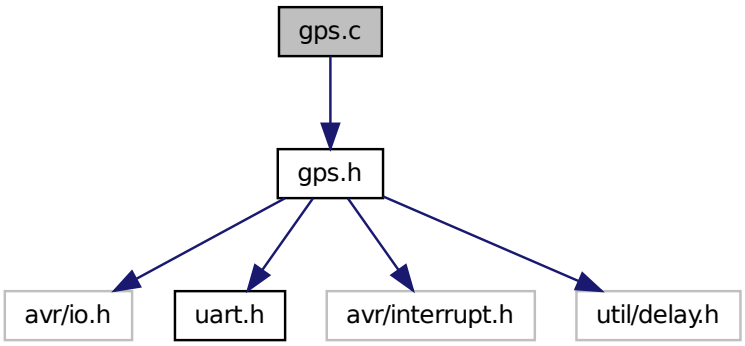
Definition at line 113 of file fat16.h.

Referenced by AppendCluster(), fgetchar_(), fopen_(), fputchar_(), fseek_(), GetFatClusterOffset(), GetNext↩ Cluster(), InitFat16(), and SeekDirectoryEntry().

7.4 gps.c File Reference

Kommunikation zwischen AVR und Navilock GPS Modul.

```
#include "gps.h"
```
Include dependency graph for gps.c:



Functions

- void gps_init ()

  Routine zur Initialisierung des GPS Moduls.
- char gps_get_char ()

  Gibt sobald verfügbar ein Zeichen vom UART zurück.
- void gps_get_nmea (char ∗buf, uint8_t bufSize)

  Empfängt einen vollständige NMEA-Befehl und gibt diesen dann zurück.

7.4.1 Detailed Description

Kommunikation zwischen AVR und Navilock GPS Modul.

Author

Martin Matysiak (mail@k621.de)
V. Pippan (webmaster@vpippan.at)

Date

2013-10-18

Version

20131018

Bei Fragen und Verbesserungen wendet euch per EMail an mich.

Datenblätter zum Umgang mit Navilock GPS Modulen:

Datenblatt Navilock U-Blox
NMEA Reference Manual

This file is part of ATmega8 GPS Logger.

ATmega8 GPS Logger is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or any later version.

ATmega8 GPS Logger is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with ATtiny2313 Cessna Lichtsteuerung. If not, see
www.gnu.org/licenses.

Definition in file gps.c.

7.4.2 Function Documentation

7.4.2.1 char gps_get_char (void)

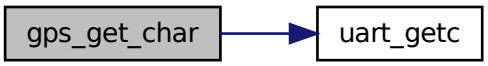
Gibt sobald verfügbar ein Zeichen vom UART zurück.

Definition at line 53 of file gps.c.

References UART_BUFFER_OVERFLOW, uart_getc(), and UART_NO_DATA.

Referenced by gps_get_nmea().

Here is the call graph for this function:



Here is the caller graph for this function:

7.4.2.2 void gps_get_nmea (char * *buf*, uint8_t *bufSize*)

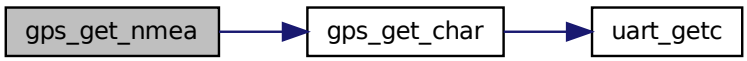
Empfängt einen vollständige NMEA-Befehl und gibt diesen dann zurück.

Definition at line 69 of file gps.c.

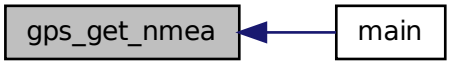
References gps_get_char().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.2.3 void gps_init (void)

Routine zur Initialisierung des GPS Moduls.

Definition at line 41 of file gps.c.

References F_CPU, GPS_BAUD, UART_BAUD_SELECT, and uart_init().

Referenced by main().

Here is the call graph for this function:

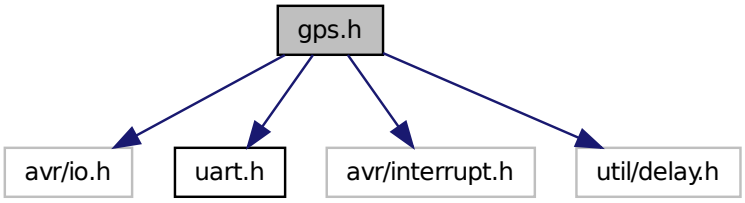
Here is the caller graph for this function:

7.5 gps.h File Reference

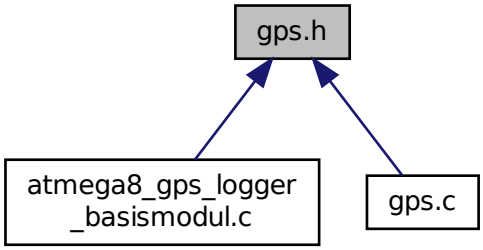
Include File für gps.c.

```
#include <avr/io.h>
#include "uart.h"
#include <avr/interrupt.h>
#include <util/delay.h>
```

Include dependency graph for gps.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define GPS_BAUD 4800

Functions

- void gps_init (void)

  Routine zur Initialisierung des GPS Moduls.
- char gps_get_char (void)

  Gibt sobald verfügbar ein Zeichen vom UART zurück.
- void gps_get_nmea (char *buf, uint8_t bufSize)

  Empfängt einen vollständige NMEA-Befehl und gibt diesen dann zurück.

7.5.1 Detailed Description

Include File für gps.c.

Author

Martin Matysiak (mail@k621.de)
Pippan Vincent (webmaster@vpippan.at)

Date

2013-10-18

Version

20131018

Definition in file gps.h.

7.5.2 Macro Definition Documentation

7.5.2.1 #define GPS_BAUD 4800

Definition at line 18 of file gps.h.

Referenced by gps_init().

7.5.3 Function Documentation

7.5.3.1 char gps_get_char (void)

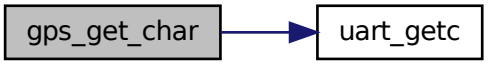
Gibt sobald verfügbar ein Zeichen vom UART zurück.

Definition at line 53 of file gps.c.

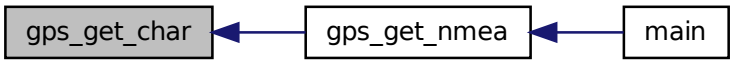
References UART_BUFFER_OVERFLOW, uart_getc(), and UART_NO_DATA.

Referenced by gps_get_nmea().

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.2 void gps_get_nmea (char * *buf*, uint8_t *bufSize*)

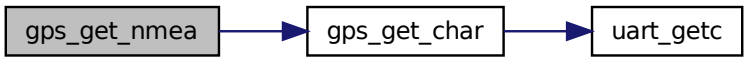
Empfängt einen vollständige NMEA-Befehl und gibt diesen dann zurück.

Definition at line 69 of file gps.c.

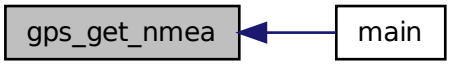
References gps_get_char().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.3 void gps_init (void)

Routine zur Initialisierung des GPS Moduls.

Definition at line 41 of file gps.c.

References F_CPU, GPS_BAUD, UART_BAUD_SELECT, and uart_init().

Referenced by main().

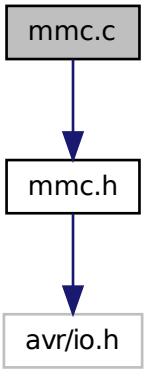
Here is the call graph for this function:

Here is the caller graph for this function:

7.6 mmc.c File Reference

```
#include "mmc.h"
```

Include dependency graph for mmc.c:



Functions

- unsigned char mmc_init ()
- unsigned char mmc_write_command (unsigned char ∗cmd)
- unsigned char mmc_read_byte (void)
- void mmc_write_byte (unsigned char Byte)
- unsigned char mmc_write_sector (unsigned long addr, unsigned char ∗Buffer)
- void mmc_read_block (unsigned char ∗cmd, unsigned char ∗Buffer, unsigned int Bytes)
- unsigned char mmc_read_sector (unsigned long addr, unsigned char ∗Buffer)
- unsigned char mmc_read_cid (unsigned char ∗Buffer)
- unsigned char mmc_read_csd (unsigned char ∗Buffer)

7.6.1 Function Documentation

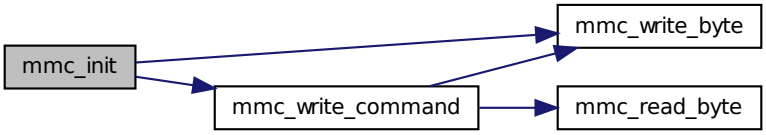
7.6.1.1 unsigned char mmc_init (void)

Definition at line 32 of file mmc.c.

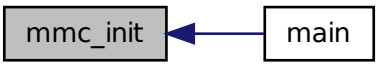
References MMC_Direction_REG, MMC_Disable, MMC_Write, mmc_write_byte(), mmc_write_command(), and nop.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



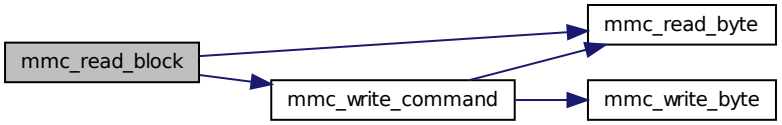
7.6.1.2 void mmc_read_block (unsigned char ∗ *cmd,* unsigned char ∗ *Buffer,* unsigned int *Bytes*)

Definition at line 254 of file mmc.c.

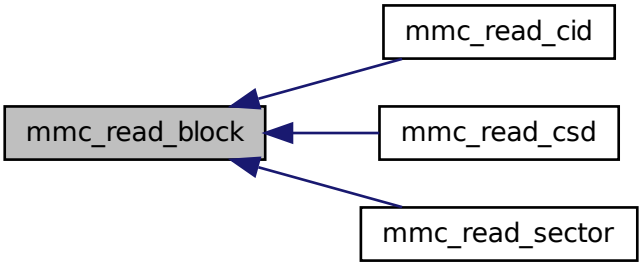
References MMC_Disable, mmc_read_byte(), and mmc_write_command().

Referenced by mmc_read_cid(), mmc_read_csd(), and mmc_read_sector().

Here is the call graph for this function:



Here is the caller graph for this function:



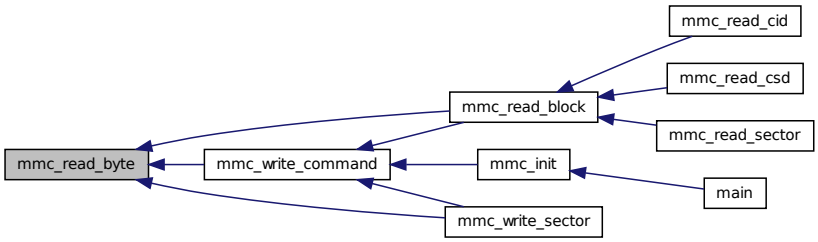
7.6.1.3 unsigned char mmc_read_byte (void)

Definition at line 135 of file mmc.c.

References MMC_Read, and MMC_Write.

Referenced by mmc_read_block(), mmc_write_command(), and mmc_write_sector().

Here is the caller graph for this function:

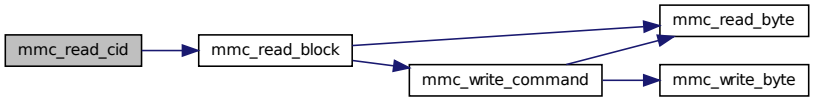


7.6.1.4 unsigned char mmc_read_cid (unsigned char * *Buffer*)

Definition at line 309 of file mmc.c.

References mmc_read_block().

Here is the call graph for this function:

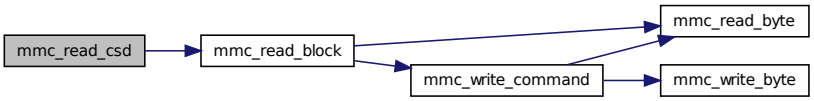


7.6.1.5 unsigned char mmc_read_csd (unsigned char * *Buffer*)

Definition at line 322 of file mmc.c.

References mmc_read_block().

Here is the call graph for this function:

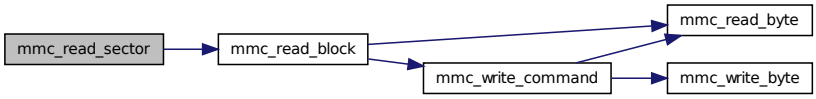


7.6.1.6 unsigned char mmc_read_sector (unsigned long *addr,* unsigned char * *Buffer*)

Definition at line 286 of file mmc.c.

References mmc_read_block().

Here is the call graph for this function:



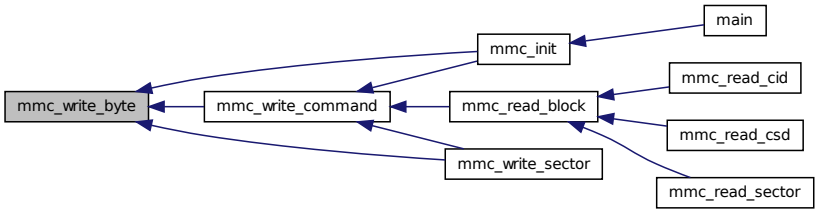
7.6.1.7 void mmc_write_byte (unsigned char *Byte*)

Definition at line 166 of file mmc.c.

References MMC_Write.

Referenced by mmc_init(), mmc_write_command(), and mmc_write_sector().

Here is the caller graph for this function:



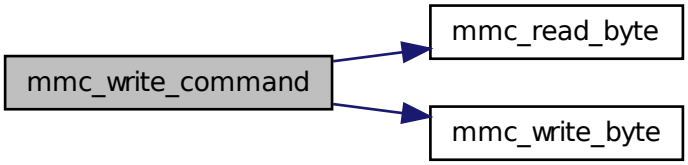
7.6.1.8 unsigned char mmc_write_command (unsigned char * *cmd*)

Definition at line 99 of file mmc.c.

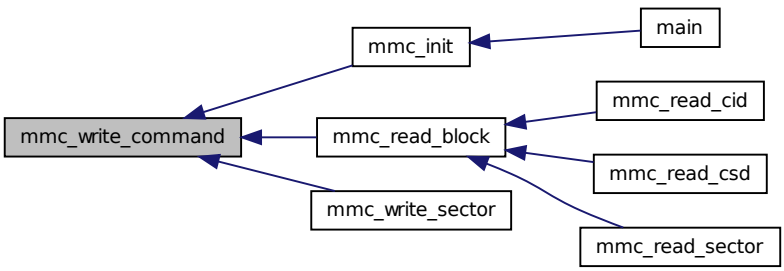
References MMC_Disable, MMC_Enable, mmc_read_byte(), and mmc_write_byte().

Referenced by mmc_init(), mmc_read_block(), and mmc_write_sector().

Here is the call graph for this function:



Here is the caller graph for this function:

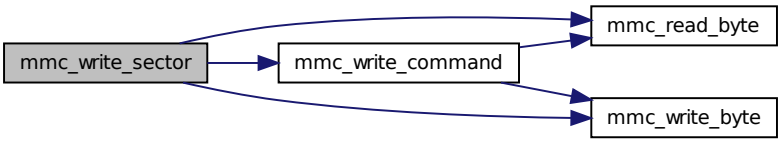


7.6.1.9 unsigned char mmc_write_sector (unsigned long *addr,* unsigned char * *Buffer*)

Definition at line 195 of file mmc.c.

References MMC_Disable, mmc_read_byte(), mmc_write_byte(), and mmc_write_command().

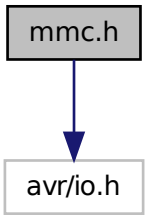
Here is the call graph for this function:



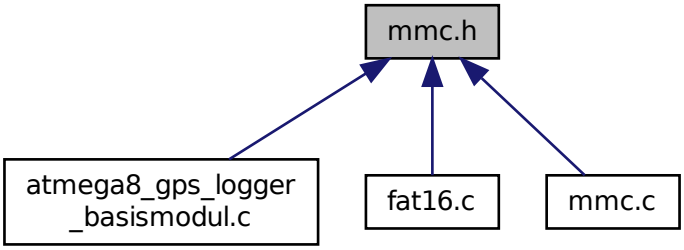
7.7 mmc.h File Reference

```
#include <avr/io.h>
```

Include dependency graph for mmc.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define SPI_Mode 1
- #define MMC_Write PORTB
- #define MMC_Read PINB
- #define MMC_Direction_REG DDRB
- #define MMC_Disable() MMC_Write|= (1<<MMC_Chip_Select);
- #define MMC_Enable() MMC_Write&=~(1<<MMC_Chip_Select);
- #define SDC_PutSector mmc_write_sector
- #define SDC_GetSector mmc_read_sector
- #define nop() __asm__ __volatile__ ("nop" ::)

Functions

- unsigned char mmc_read_byte (void)
- void mmc_write_byte (unsigned char)
- void mmc_read_block (unsigned char *, unsigned char *, unsigned in)
- unsigned char mmc_init (void)
- unsigned char mmc_read_sector (unsigned long, unsigned char *)
- unsigned char mmc_write_sector (unsigned long, unsigned char *)
- unsigned char mmc_write_command (unsigned char *)
- unsigned char mmc_read_csd (unsigned char *)
- unsigned char mmc_read_cid (unsigned char *)

7.7.1 Macro Definition Documentation

7.7.1.1 #define MMC_Direction_REG DDRB

Definition at line 17 of file mmc.h.

Referenced by mmc_init().

7.7.1.2 #define MMC_Disable( ) MMC_Write|= (1<<MMC_Chip_Select);

Definition at line 63 of file mmc.h.

Referenced by mmc_init(), mmc_read_block(), mmc_write_command(), and mmc_write_sector().

7.7.1.3 #define MMC_Enable( ) MMC_Write&=~(1<<MMC_Chip_Select);

Definition at line 66 of file mmc.h.

Referenced by mmc_write_command().

7.7.1.4 #define MMC_Read PINB

Definition at line 16 of file mmc.h.

Referenced by mmc_read_byte().

7.7.1.5 #define MMC_Write PORTB

Definition at line 15 of file mmc.h.

Referenced by mmc_init(), mmc_read_byte(), and mmc_write_byte().

7.7.1.6 #define nop( ) __asm__ __volatile__ ("nop" ::)

Definition at line 72 of file mmc.h.

Referenced by mmc_init().

7.7.1.7 #define SDC_GetSector mmc_read_sector

Definition at line 70 of file mmc.h.

Referenced by AppendCluster(), CreateDirectoryEntry(), fflush_(), fgetchar_(), FindNextFreeCluster(), fseek_(), GetNextCluster(), InitFat16(), and SeekDirectoryEntry().

7.7.1.8 #define SDC_PutSector mmc_write_sector

Definition at line 68 of file mmc.h.

Referenced by AppendCluster(), CreateDirectoryEntry(), fflush_(), FindNextFreeCluster(), and fputchar_().

7.7.1.9 #define SPI_Mode 1

Definition at line 12 of file mmc.h.

7.7.2 Function Documentation

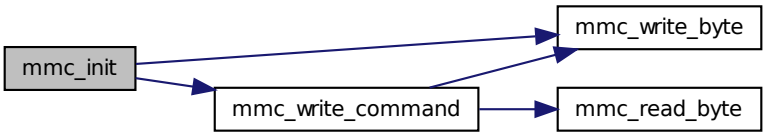
7.7.2.1 unsigned char mmc_init (void)

Definition at line 32 of file mmc.c.

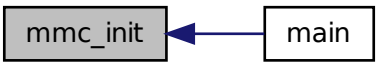
References MMC_Direction_REG, MMC_Disable, MMC_Write, mmc_write_byte(), mmc_write_command(), and nop.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.2.2 void mmc_read_block (unsigned char *, unsigned char *, unsigned *in*)

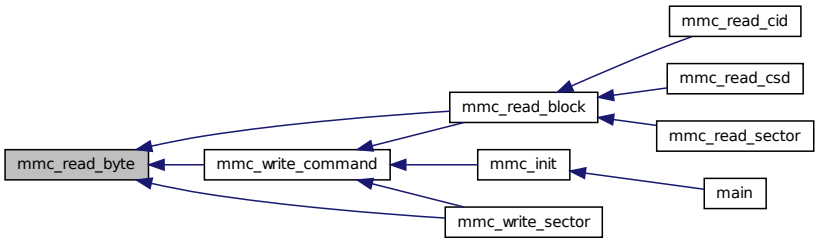
7.7.2.3 unsigned char mmc_read_byte (void)

Definition at line 135 of file mmc.c.

References MMC_Read, and MMC_Write.

Referenced by mmc_read_block(), mmc_write_command(), and mmc_write_sector().

Here is the caller graph for this function:

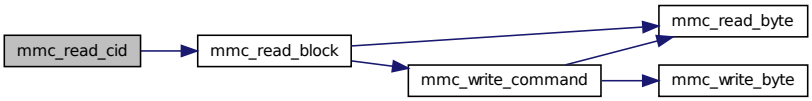


7.7.2.4 unsigned char mmc_read_cid (unsigned char *)

Definition at line 309 of file mmc.c.

References mmc_read_block().

Here is the call graph for this function:

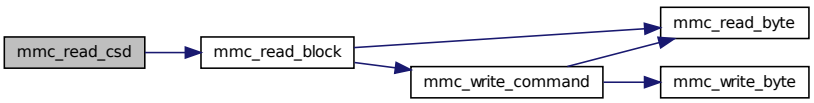


7.7.2.5 unsigned char mmc_read_csd (unsigned char *)

Definition at line 322 of file mmc.c.

References mmc_read_block().

Here is the call graph for this function:

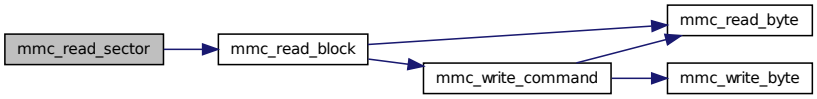


7.7.2.6 unsigned char mmc_read_sector (unsigned *long,* unsigned char *)

Definition at line 286 of file mmc.c.

References mmc_read_block().

Here is the call graph for this function:



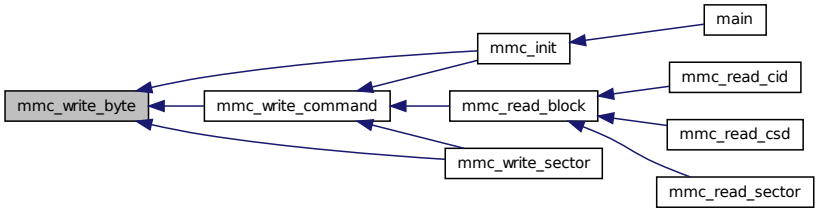
7.7.2.7 void mmc_write_byte (unsigned *char*)

Definition at line 166 of file mmc.c.

References MMC_Write.

Referenced by mmc_init(), mmc_write_command(), and mmc_write_sector().

Here is the caller graph for this function:



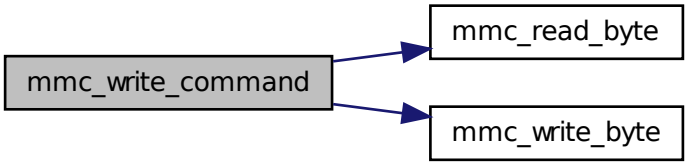
7.7.2.8 unsigned char mmc_write_command (unsigned char *)

Definition at line 99 of file mmc.c.

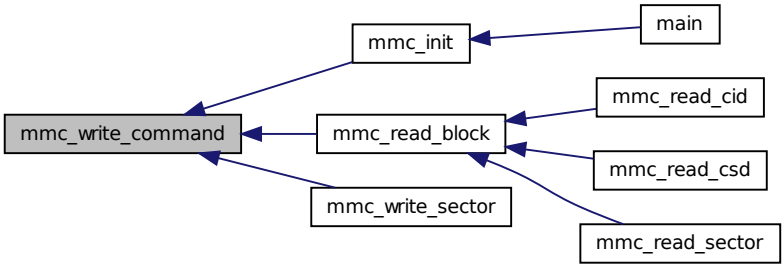
References MMC_Disable, MMC_Enable, mmc_read_byte(), and mmc_write_byte().

Referenced by mmc_init(), mmc_read_block(), and mmc_write_sector().

Here is the call graph for this function:



Here is the caller graph for this function:

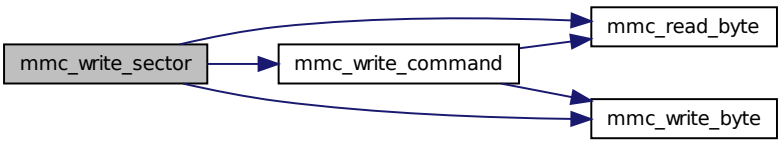


7.7.2.9 unsigned char mmc_write_sector (unsigned *long,* unsigned char *)

Definition at line 195 of file mmc.c.

References MMC_Disable, mmc_read_byte(), mmc_write_byte(), and mmc_write_command().

Here is the call graph for this function:

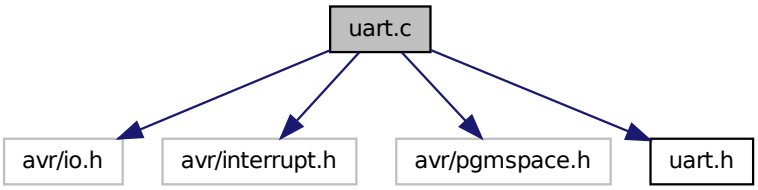


7.8 uart.c File Reference

Interrupt UART library with receive/transmit circular buffers.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/pgmspace.h>
#include "uart.h"
```

Include dependency graph for uart.c:



Macros

- #define UART_RX_BUFFER_MASK (UART_RX_BUFFER_SIZE - 1)
- #define UART_TX_BUFFER_MASK (UART_TX_BUFFER_SIZE - 1)

Functions

- SIGNAL (UART0_RECEIVE_INTERRUPT)
- SIGNAL (UART0_TRANSMIT_INTERRUPT)
- void uart_init (unsigned int baudrate)

  Initialize UART and set baudrate.
- unsigned int uart_getc (void)

  Get received byte from ringbuffer.
- void uart_putc (unsigned char data)

  Put byte to ringbuffer for transmitting via UART.
- void uart_puts (const char *s)

  Put string to ringbuffer for transmitting via UART.
- void uart_puts_p (const char *progmem_s)

  Put string from program memory to ringbuffer for transmitting via UART.

7.8.1 Detailed Description

Interrupt UART library with receive/transmit circular buffers.

Author

Peter Fleury <pfleury@gmx.ch>
Peter Fleury

Date

2007-07-01

Version

1.6.2.1

Software: AVR-GCC 4.1, AVR Libc 1.4.6 or higher
Hardware: any AVR with built-in UART
License: GNU General Public License

DESCRIPTION:
An interrupt is generated when the UART has finished transmitting or receiving a byte. The interrupt handling routines use circular buffers for buffering received and transmitted data.

The UART_RX_BUFFER_SIZE and UART_TX_BUFFER_SIZE variables define the buffer size in bytes. Note that these variables must be a power of 2.

USAGE:
Refere to the header file uart.h for a description of the routines. See also example test_uart.c.

Note

Based on Atmel Application Note AVR306

Definition in file uart.c.

7.8.2 Macro Definition Documentation

7.8.2.1 #define UART_RX_BUFFER_MASK (UART_RX_BUFFER_SIZE - 1)

constants and macros

size of RX/TX buffers

Definition at line 52 of file uart.c.

Referenced by SIGNAL(), and uart_getc().

7.8.2.2 #define UART_TX_BUFFER_MASK (UART_TX_BUFFER_SIZE - 1)

Definition at line 53 of file uart.c.

Referenced by SIGNAL(), and uart_putc().

7.8.3 Function Documentation

7.8.3.1 SIGNAL (UART0_RECEIVE_INTERRUPT)

UART Receive Complete interrupt called when the UART has received a character read UART status register and UART data register

calculate buffer index

error: receive buffer overflow

store new index

store received data in buffer

Definition at line 248 of file uart.c.

References UART_BUFFER_OVERFLOW, and UART_RX_BUFFER_MASK.

7.8.3.2 SIGNAL (UART0_TRANSMIT_INTERRUPT)

UART Data Register Empty interrupt called when the UART is ready to transmit the next byte calculate and store new buffer index

get one byte from buffer and write it to UART

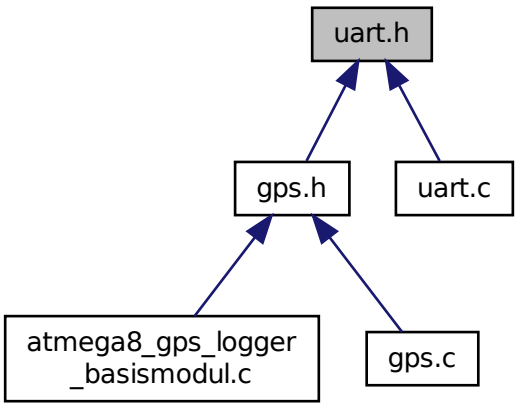
tx buffer empty, disable UDRE interrupt

Definition at line 290 of file uart.c.

References UART_TX_BUFFER_MASK.

7.9 uart.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define UART_BAUD_SELECT(baudRate, xtalCpu) ((xtalCpu)/((baudRate)∗16l)-1)

  UART Baudrate Expression.
- #define UART_BAUD_SELECT_DOUBLE_SPEED(baudRate, xtalCpu) (((xtalCpu)/((baudRate)∗8l)-1)|0x8000)

  UART Baudrate Expression for ATmega double speed mode.
- #define UART_RX_BUFFER_SIZE 64
- #define UART_TX_BUFFER_SIZE 64

- #define UART_FRAME_ERROR 0x0800 /* Framing Error by UART */
- #define UART_OVERRUN_ERROR 0x0400 /* Overrun condition by UART */
- #define UART_BUFFER_OVERFLOW 0x0200 /* receive ringbuffer overflow */
- #define UART_NO_DATA 0x0100 /* no receive data available */
- #define uart_puts_P(__s) uart_puts_p(PSTR(__s))

  Macro to automatically put a string constant into program memory.
- #define uart1_puts_P(__s) uart1_puts_p(PSTR(__s))

  Macro to automatically put a string constant into program memory.

Functions

- void uart_init (unsigned int baudrate)

  Initialize UART and set baudrate.
- unsigned int uart_getc (void)

  Get received byte from ringbuffer.
- void uart_putc (unsigned char data)

  Put byte to ringbuffer for transmitting via UART.
- void uart_puts (const char *s)

  Put string to ringbuffer for transmitting via UART.
- void uart_puts_p (const char *progmem_s)

  Put string from program memory to ringbuffer for transmitting via UART.
- void uart1_init (unsigned int baudrate)

  Initialize USART1 (only available on selected ATmegas)
- unsigned int uart1_getc (void)

  Get received byte of USART1 from ringbuffer. (only available on selected ATmega)
- void uart1_putc (unsigned char data)

  Put byte to ringbuffer for transmitting via USART1 (only available on selected ATmega)
- void uart1_puts (const char *s)

  Put string to ringbuffer for transmitting via USART1 (only available on selected ATmega)
- void uart1_puts_p (const char *s)

  Put string from program memory to ringbuffer for transmitting via USART1 (only available on selected ATmega)

Index