UNIVERSIDAD
D
CORDOBA

INSTITUTO DE ESTUDIOS DE POSTGRADO
MÁSTER EN ENERGÍAS RENOVABLES DISTRIBUIDAS

COST EFFECTIVE TECHNOLOGY APPLIED TO DOMOTICS AND SMART HOME ENERGY MANAGEMENT SYSTEMS

JAVIER ALBERTO GUTIÉRREZ PEÑA

Tesis presentada a la Comisión integrada por los profesores:

DR. D. JOSÉ MARÍA FLORES ARIAS

DR. D. FRANCISCO JOSÉ BELLIDO OUTEIRIÑO

Para completar las exigencias del Máster
en Energías Renovables Distribuidas

Córdoba, septiembre, 2020

DEDICATORIA

A mi mujer, hijos, mis hermanos y mis padres sin cuya comprensión no habría podido compaginar la realización de este trabajo con mi vida profesional y personal.

AGRADECIMIENTOS

Agradezco a Eva María y a Javier y Juan Carlos, por el tiempo que debería haberles dedicado y en su lugar he empleado en la realización de este trabajo y en el máster.

A Rafael por la ayuda activa y el tiempo que me ha prestado para poder concluir este trabajo. A Pedro Antonio y a Charo por sus ánimos y apoyo, moral e intelectual.

A Chari y a Rafael por TODO, pues ellos me lo han dado todo, todo lo que soy; por encima de todo por los valores y educación que me han convertido en lo que soy.

A Juan Carlos, por su colaboración y acompañamiento durante esta aventura del saber.

Y a Francisco José y José María por su guía, paciencia y consejos que me han permitido mejorar en la realización de este documento; a ellos de nuevo y al resto de profesores que imparten este máster, que me ha permitido adquirir los conocimientos necesarios, parte de los cuales espero haber reflejado en este trabajo al nivel al que ellos supieron transmitirla.

GENERAL INDEX

TABLES INDEX

LIST OF FIGURES

RESUMEN

En este documento se aborda el estado actual de la domótica de bajo coste disponible en el mercado actualmente y cómo aplicarlo en los sistemas inteligentes de gestión energética en la vivienda (SHEMS) permitiendo el recorte de las puntas de demanda, gestión de energías renovables y control de electrodomésticos, siempre en el contexto del bajo coste, con el objetivo de lograr la máxima difusión de los SHEMS. Adicionalmente, más allá del contexto de la tecnología SHEMS, se analizará cómo aplicar esta tecnología para aumentar la eficiencia energética de los hogares y para la supervisión de los electrodomésticos.

La gestión energética es uno de los factores principales para lograr la difusión de las energías renovables distribuidas; debido a que las fuentes de energía renovable no pueden ser planificadas, se requieren sistemas de control capaces de gestionar el intercambio de energía entre las fuentes convencionales (red eléctrica de distribución), energías renovables y dispositivos de almacenamiento energético.

Bajo esta perspectiva, este documento presenta un primer bloque en el que se exponen las bases de la arquitectura y módulos funcionales de los sistemas inteligentes de gestión energética en la vivienda (SHEMS); el siguiente paso será analizar los principios que han permitido a ciertos dispositivos convertirse en dispositivos de bajo coste.

Una vez analizada la tecnología, nos centraremos en los recursos (hardware y software) existentes que permitirán la realización de un SHEMS a bajo coste.

Conocidas las “herramientas” a nuestra disposición, se mostrará como adaptar un esquema SHEMS clásico a la tecnología de bajo coste. Primeramente, comparando de modo genérico la tecnología de bajo coste con una de las principales propuestas comerciales de SHEMS, para seguidamente desarrollar la solución de bajo coste a un caso específico real.

Palabras Claves: Domótica, bajo coste, SHEMS, Sistemas inteligentes de gestión energética en la vivienda, IoT, Internet de las cosas.

ABSTRACT

In this document is presented the state of art for domotics cost effective technologies available on market nowadays, and how to apply them in Smart Home Energy Management Systems (SHEMS) allowing peaks shaving, renewable management and home appliance controls, always in cost effective context in order to be massively applied. Additionally, beyond of SHEMS context, it will be also analysed how to apply this technology in order to increase homes energy efficiency and monitoring of home appliances.

Energy management is one of the milestones for distributed renewable energy spread; since renewable energy sources are not time-schedulable, are required control systems capable of the management for exchanging energy between conventional sources (power grid), renewable sources and energy storage sources.

With the proposed approach, there is a first block dedicated to show an overview of Smart Home Energy Management Systems (SMHEMS) classical architecture and functional modules of SHEMS; next step is to analyse principles which has allowed some devices to become a cost-effective technology.

Once the technology has been analysed, it will be reviewed some specific resources (hardware and software) available on marked for allowing low cost SHEMS.

Knowing the "tools" available; it will be shown how to adapt classical SHEMS to cost effective technology. Such way, this document will show some specific applications of SHEMS. Firstly, in a general point of view, comparing the proposed low-cost technology with one of the main existing commercial proposals; and secondly, developing the solution for a specific real case.

Keywords: Domotics, cost effective, SHEMS, Smart Home Energy Management System, IoT.

1. INTRODUCTION

Distributed renewable energy (DRE) is based on energy generation at the same point or as close as possible where it will be consumed. DRE additional to an alternative for increasing renewable energies use are an unprecedented opportunity to accelerate the transition to modern energy services in remote and rural areas [1].
Implementing DRE systems in home, consumers become producers (prosumers). Usually, small energy store systems (ESS) are added to the individual cell (home), becoming in Prosumage:

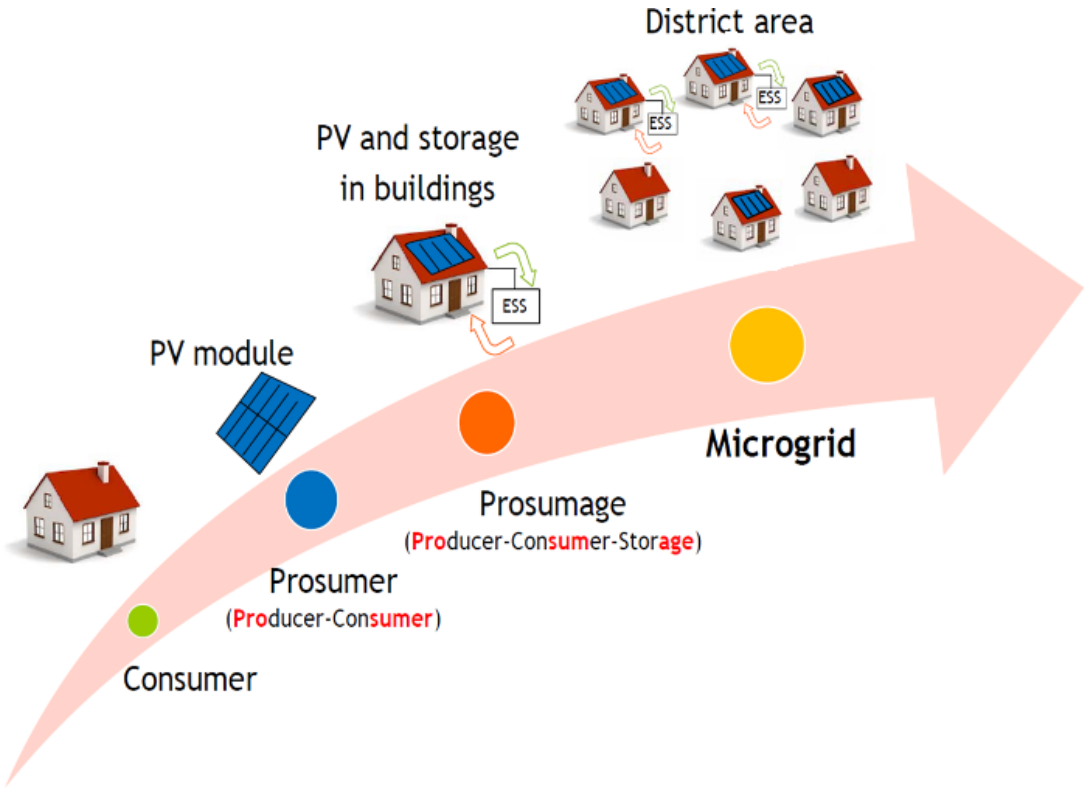


Figure 1 - From consumer to microgrid evolution [2].

Finally, when a cluster of prosumages is created it can be considered a microgrid.
But one step before conforming the microgrid, each individual cell, each home, must be capable of combining the three roles they are playing, individually. To get the higher efficiency possible a control system must be designed in order to synchronize those different roles in the best way possible, assuming the specifications for each of those roles:

- Consumers: they are connected to the wide traditional distribution grid. They must be synchronized to external grid; consuming of it the less possible in one hand, but securing the electrical supply to home, in the other hand. When it is required to consume, it must be at the lower cost possible; electrical companies offer the lower rates at peak-off periods, so, consume when the power consume is lower has a “premium” and our control system must take advantage of it. But it is not just the connection to external wide grid what needs to be

controlled. Also, each energy load, inside the home must be controlled. Each load will have some requirements: ones can be delayed and scheduled in time while others cannot (e.g. washing machine vs air conditioning); ones are critical, and others are not (e.g. fridge vs TV). Further than that, using a control system some consume patterns can be adapted to the most effective way, e.g. air conditioning can start to work before people arrives home, so that the house is over cooled in order to avoid power consumption during costly hours. And finally, home energy efficiency can be increased using the control system: in summer, activating sunshades automatically when the sun is up, and opening windows blinds when the day is cooler or regulating heaters temperature according to external temperature forecast.

- Producers: Each home can be equipped with different renewable energy sources, like photovoltaic solar and/or wind power (new technology Vortex Bladeless © [3] is allowing domestic wind energy production) and/or geothermal energy and/or thermal solar. Each of those sources has their own characteristics, but generally they are no stable, because they depend on external elements. Most of them cannot be accurately scheduled, just approximately using weather forecast. Control system must coordinate such energy sources dynamically with power consume, trying the get the best use of renewable energy, when the energy is produced, adjusting loads to the availability of those sources.
- Energy storage: Though it is not mandatory in this model, many houses can be equipped with ESS (Energy Storage Systems) which allow them to storage energy when it is more affordable and use it when it is more needed. So, when external energy is cheaper or when renewable energy is being generated but not consumed it can be stored; and it can be used when external energy is expensive or when renewable energy is not available. Also, those storage systems can have some requirements related to charging and discharging cycles. Control system must command when energy storage must be started, when must be stopped and when it must be used.

Finally, advantage can be taken on the fact that there is a network of appliances interconnected to central unit smart controller in order to develop not DRE related functions such as:

- Obtaining power consumption appliances fingerprint in order to detect deviations over the original; that deviation will indicate a wrong function before further damage is produced so they are easy to fix, and unplanned outage is avoided
- Remote start up or shut down of devices, allowing conditioning the house before arriving and/or life simulation when house is empty.
- Using voice or even just detecting presence at home (using, e.g. mobile connection to home Wi-Fi) for activating a system of devices so that when someone arrives home control system will turn on the lights, start heating systems and turn on oven for preparing dinner.
- …

Anyway, above examples are out of this work scope, more focused to DRE than domotics; nonetheless, this possibility is pointed in order to remark the cost-effective perspective of the technology analysed, being able to be used beyond that a DRE context.

2. PREVIOUS CONCEPT - SHEMS OVERVIEW

At this chapter, a brief overview of classical architecture and functional modules of Smart Home Energy Management Systems is going to be carried on.
Advanced Metering Infrastructure (AMI) refers to the system composed of all devices, on both parts, consumers and utilities, which allows producers having exact data of power consumption on the demand side and consumers receive information from producers to adapt their loads to utilities requirements to get advantage of incentives in form of electricity prices and improvement in utilization efficiency of household appliances and residential energy conservation.
Hence, Smart Home Energy Management System (SHEMS) is defined as the optimal system providing energy management services in order to efficiently monitor and manage electricity generation, storage, and consumption in smart houses [4].
Sensing and actuating devices send and receive information to/from a central processor unit through the Home Area Network (HAN). Besides, SHEMS can provide not only the optimum utilization status of home appliances, but also energy storage and management services for distributed energy resources (DERs) and Home Energy Storage System (HESS) [5].

2.1 SHEMS BASIC ARCHITECTURE

Bellow picture is showing an overall architecture of a representative SHEMS:

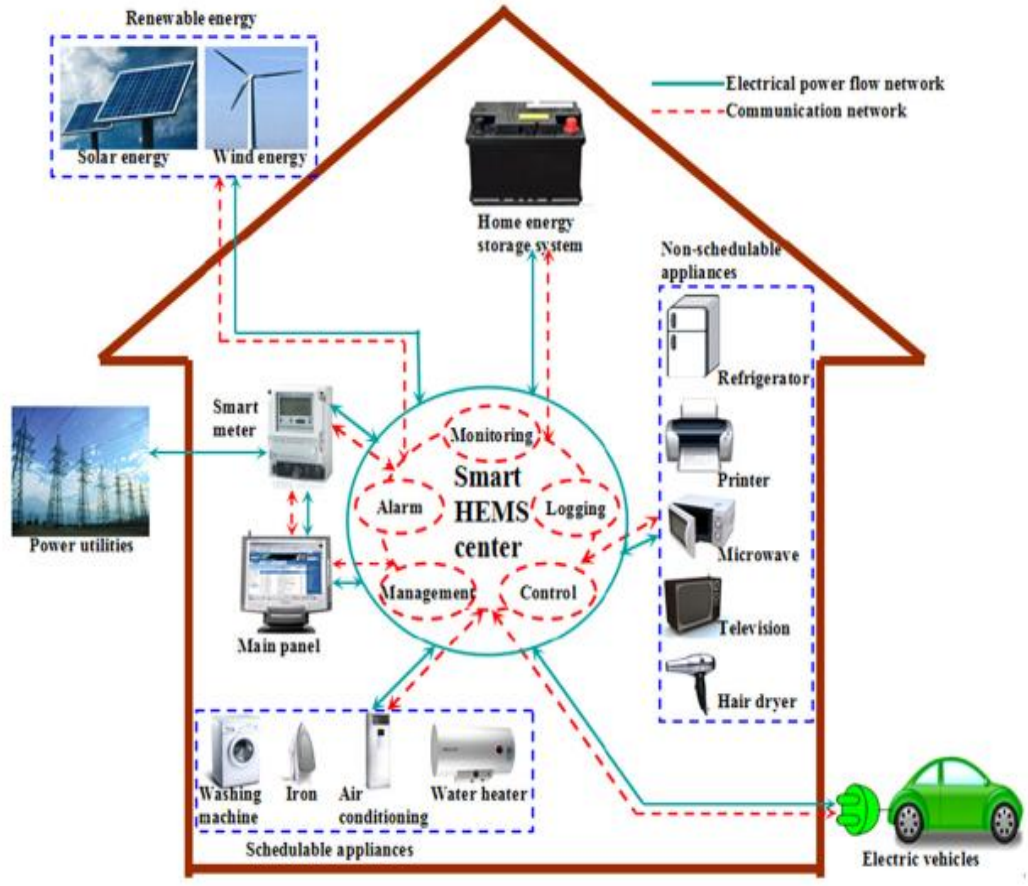


Figure 2 - Overall architecture of a representative SHEMS [7].

The central processor unit is called Centralized Smart Controller and provides to user with monitoring and control functionalities based on HAN.
SHEMS is collecting power consumption details from home and can communicate it to power utilities through the house gateway, which typically will be the smart meter device, owned by electric company. This house gateway can receive a demand response signal from power utilities as an input to SHEMS, and the optimization of home appliance scheduling can be implemented for the residential demand response [6].
Renewable energies are implemented in individual cells, so SHEMS must manage also their unpredictable profile; to develop this function takes relevance the electric vehicles, since EV are not only a load during charging, but also can be used as an storage mean, so that not consumed energy can be stored and used when renewable energy is not available. In this way consumers have become "prosumages", consuming, producing and storing energy.
Users can choose Human Machine Interface adapted to their specifications in order to schedule the different appliances service time or prioritize the different loads to be managed by SHEMS.
Basic functionalities provided by SHEMS system are:

- Monitoring: Offers easy access to real-time information on energy consumption and allows users to focus on the electricity saving. It can also provide display services for the operational modes and energy status of each home appliance.
    - Premise energy status display service.
    - Operation condition monitoring service.
- Logging: To collate and save the data information on the amount of electricity usage from appliances, generations from DERs and energy storage state. This service also contains demand response analysis for real-time prices from grid utility.
    - Energy information system.
    - Real time pricing related service.
- Control: There are two types of control, namely, direct control and remote control. Direct control is implemented on both the equipment and control system; whereas, remote control means customers can online access to monitor and control the usage patterns of in-home devices via handheld personal computer or smart phone from outside.
    - Remote monitoring and control sservice using handheld device.
- Alarm: Alarm will be generated and sent to the SHEMS center with information on the fault locations, for example, if there is any abnormality detected.
    - Home grid alarm service.
- Management: is the most important function of SHEMS to enhance the optimization and efficiency of electrical power usage in smart house. It covers a range of services including renewable energy system management service, energy storage management service, home appliance management service, and Plug-in EV and battery management service.
    - Renewable energy system management service.
    - Energy storage management service.
    - Home appliance management service.
    - Plug-in electric vehicle and battery management service.

2.2. SHEMS Infrastructure

Based on previously defined architecture, definition of the infrastructure of a generic SHEMS system is going to be develop, defining the following elements:

- SHEMS centre. It is the "core" of SHEMS corresponding to a Central Processor Unit, such a PC/barebone computer (Windows, iOS or Linux OS), PLC, but also a smart phone

though it must be considered that SHEMS centre needs to be permanently on home, tablet, Raspberry, … Main functions of SHEMS centre [8]:

- Receiving a large amount of data sent by smart meters, main control panel, and real-time display. The control commands issued by the consumer are sent to all household equipment. Consequently, the automated demand response can be achieved;
- Providing a friendly human-machine interface and supporting user's real-time browsing, online monitoring, task setting and other functions to arrange the usage of electricity;
- With high scalability, the smart HEMS centre can set water, electricity, gas, and other indoor controls;
- Integrating DERs, energy storage devices and electricity regulator of EVs as well as analysing and forecasting distributed generations to achieve the optimal control of DERs.
- Developing control functions according to classical control schemes but also algorithms of fuzzy logic, neural network, genetic algorithm, … or any other AI techniques.

- Smart meters: They allow two-direction communication; this is used, in one sense to send customer consumption and, in the other sense to provide information to consumers from power utilities and in such way consumers can optimize when turn on/off home appliances and regulate them and how to manage Distributed Energy Resources and Home Energy Storage System. Main functions for Smart Meters are [9]:
  - Measuring the multi-period and multi-mode power rates of active and reactive energy metering usage;
  - Supporting duplex (two-way) communication, sending data information and accepting instruction information, such as real-time information query, real-time electricity standard rates, meter upgrade program settings, etc.
  - Enabling the response in terms of the requirements to achieve smart load shedding and cooperating with smart meter and smart interactive terminals during the islanding transition when a failure happens on the main power grid;
  - Collecting data with smart gas meters, water meters and other versatile value-added services.
- Communication and networking systems: There are different options for implementing the required communication between the different architecture elements:
  - Power Line Communication (PLC): This system takes advantage of the already laid power lines existing at home, introducing a wide band in the power network, splitting the network in power band and wide band (digital signal) by using frequency filters. Such filters are used too for avoiding digital signal going outside of home environment. For home applications, carrier signal is set between 20 and 200 kHz. But the main application nowadays in HEMS for PLC communication is for allowing smart meters two way communication between users and electric utilities; this application is called Broad Band Over Power Line (BPL, IEEE 1901-2010), being BPL Opera the wider spread standard used for smart meters and load management.
  - ZigBee: It is IEE802.15.4 standard specification. MAC sublayer offers facilities that can be harnessed by upper layers to achieve the desired level of security. Main advantages of this protocol are low cost, simplicity and low power requirement (100 mW). Low power requirement implies limitation on transmission distances (10-100 meters) and low bandwidth, but this is not a problem for SHEMS where distance is limited to home area (if required further

distances, ZigBee mesh can be used) and amount of data to be transmitted is reduced (no need for a high band width. Those features have made ZigBee one of the most used protocols in home automation (and SHEMS). They can be based on mesh topologies for increasing the range, but in that case, they will need additional hardware and routers to provide user better connectivity; as such, these systems turn out to be very expensive [10]. E.g. Roomba interface works under Zigbee.

- Wi-Fi: Compared with another options, Wi-Fi is complex and requires high power; additionally, home Wi-Fi network is already congested, so that add to it more devices can be a problem. But even when Wi-Fi characteristics doesn't fit with SHEMS requirements, its huge spread and mass production has made the components less expensive than any other option in market considering Wi-Fi function already embedded in low cost Systems on a Chip (SoC) like ESP32 which is making Wi-Fi a good option for low-cost domotics. For avoiding congesting of Wi-Fi network, different solutions, as zones splitting, are present on market. Wi-Fi allows protocol MQTT, which has become the easiest way for devices communication.
- Bluetooth Low Energy: BLE has a power consume of 10 mW. Reduced range (10 m) it can be a problem, even in-home area, but latest review of Bluetooth allows Bluetooth mesh increasing the range. Components cost is the most reduced considering just communication elements (not SoC) and no mesh.
- ZWave: It consumes 1 mW, with a range of 30 meters, high scalability (above 6000 elements) and comparatively high cost. It works in 900 MHz. Each device can act as receiver and repeater. There is no interference of Wi-Fi or any other 2.4GHz wireless technology. Better interoperability than ZigBee. Complicated to work with.

- Home Appliances: There are two sides for considering home appliances: as a load (more focused on SHEMs perspective) and as a "function" to be monitor. As load perspective, they can be divided in two categories:
  - Schedulable loads, which can be scheduled for optimal operation or switched on/off at any time. E.g.: washing machine, air conditioner, iron, water heater, Electric Vehicle, … They can be sub-divided in:
    - Interruptible: More easily schedulable.
    - Non-interruptible: Are constrained by fixed operation, "hold-time". Of special interest are Electric Vehicles. Expected to be widely broadcast in next years, EV can work in both directions, charging from energy source or discharging to energy source; it is called "vehicle to grid" and it will add ESS function to a majority of homes allowing peak shaving.
  - Non-schedulable loads, those which need human control or presence or those critical that cannot be interrupted. e.g. lights, refrigerator, printer, microwave, television, hair dryer.

Attending to "function" point of view, home appliances can be categorized as:

- Smart home appliances: Those which can be directly using some type of digital communication based in some specific protocol and connected to SHEMS providing high degree of control level (regulation and others) and also providing high detail in home appliances information such as power consume, device self-diagnosis, timing, … One issue with those devices is that communication protocol uses to be proprietary, not allowing easy integration with other third-party systems. Smart lighting can be included in this group, although they are adopting also standard protocols.

- Traditional home appliances: They cannot be smartly controlled, in principle just ON/OFF. Nonetheless, they can be modified in order to provide more detailed control (e.g. connecting push buttons increase/decrease temperature in air conditioner to smart plugs).

- Smart plug: Acting as circuit breakers, to start-on/off different loads or connecting disconnecting power sources or batteries (or similar as EV).
- Smart regulators: Providing a variable electric variable (resistance, current, …) to control a device.
- Smart Sensors: Providing information such as ambient temperature, individual power consumption (and even load fingerprint), light intensity, … but also more complex sensors, like RFID for control access or even router configured to signal when some smart phone is within range.
- ESS: Energy Storage management: As already discussed, not only dedicated ESS system, but also PEV can be considered as ESS; furthermore, in terms of management, possibility of selling electric energy to grid can be also considered as ESS system, but different price rate for selling and purchasing energy must be took into account, in order to optimize energy management.

2.3. Energy Scheduling strategies SHEMS

In the deregulated energy market cost reduction has acquired critical relevance; in order to reduce costs, it is important to get the energy demand as stable as possible, so the surplus generation capability is reduced to minimum and utilization of energy transmission infrastructures is maximized. At the end of the day, this is a basic economic principle: the more use of the resource the faster payoff of the investment and the better apply of economies of scale.
In order to adapt the demand so that it fit the best efficiency of the system utilities offer better prices for those periods in which the cumulative demand is lower. It can be simplified as the offer and demand law, as greater the demand is, higher the price raise.
As commented in third paragraph, section 2.1, plug-in electric vehicles (PEV) can be considered as dynamic storage systems. As also it has been commented, home renewables systems provide alternative energy sources.
So, combining all three previous factors, different strategies can be adopted attending to different objectives:

- Optimize and implement the home the home appliance scheduling with electrical energy services for the residential consumers in smart houses [11].
- The application of PEVs as dynamic energy storages with their travel patterns to coordinate the optimal home energy scheduling in a residential community has been presented in [12], [13] and [14].
- Automatic energy consumption scheduling strategies with price predictors to minimize electricity payment in a real-time pricing tariff environment was proposed in [15] and [16]. Application example showed in section 4.2. is based on this strategy.
- Considering various uncertainties on appliance operational time, intermittent renewable generations and variations of electricity prices, the stochastic efficient scheduling schemes for optimized SHEMSs have been addressed in [17] and [18].
- Moreover, based on the bidirectional communication network architecture to schedule the in-building appliances and renewable energy sources, the distributed control algorithms for household demand response have been presented in [19].
- A multi-objective Demand Response (DR) optimization model which is formulated as a multi-objective nonlinear programming problem and subjected to a set of constraints

which and is solved using the Non-Dominated Sorted Genetic Algorithm (NSGA-II), in order to determine the scheduling of home appliances for the time horizon is presented on [20] not only in order to minimize the cost of the electricity but also minimizing the level of inconvenience for residential consumers.

- A strategy based on the modeller system Game theory working with advanced metering system (GAMS), which provides numerical programming and optimization software points, provides to energy consumption reduction by 48% and maximizes the renewable energy consumed at the rate 65% of the total energy generated as it is proposed in [21].

Besides, a smart HEMS shall be able to respond to renewable generation fluctuations, electricity price, and other human behaviour influences in real-time or near real-time to achieve a comfortable lifestyle with financial incentives [22]. The system shall also be flexible enough to accommodate and manage various home appliances, renewable energy resources and HESSs for energy saving and demand response.

All those different strategies are implemented on SHEMS centre. Since SHEMS is a programmable device CPU (from a complete desktop system to a barebone computer or Raspberry Pi), depending on power of CPU, accessory chips (video processor, sound, mathematical, …) practically, any algorithm can be implemented, using C+ code, or MATLAB routines, python, … whatever is needed; high level smart programming methods can be applied on it, such as genetics, neural networks, fuzzy logic, … can be integrated on a complex system. In conference paper [23] an approximation to such mixture of intelligence techniques is proposed to achieve a productive and cost-effective environment through optimization of air conditioning based on ambient temperature and humidity mainly, but also considering light and motion:

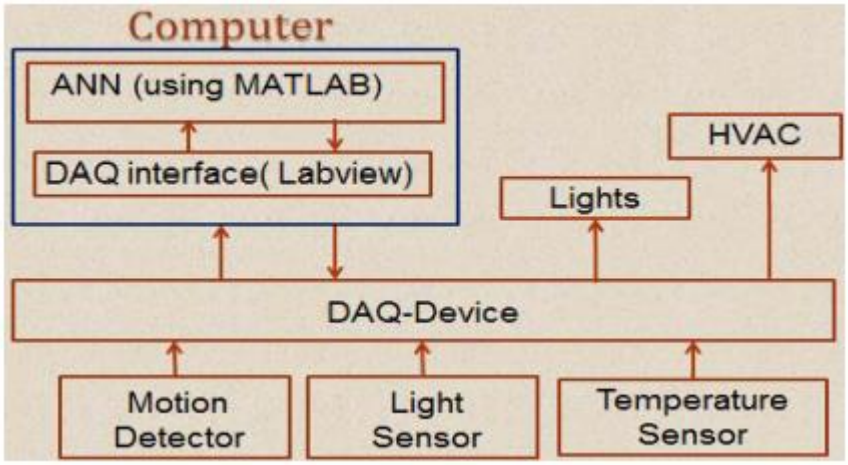


Figure 3 - Block Diagram of the Smart Home System [24]

In this approximation, LabView is used as interface between sensors/actuators and MATLAB, being MATLAB the real intelligence on this system, which firstly classify data using fuzzy logic module and secondly, taking as inputs those classified data and previous history of the controlled device, constructs an ANN model which predicts the next setting of the controlled device. Section 3.3.2.4 Thigspeak application goes deeper in Matlab application for SHEMS.

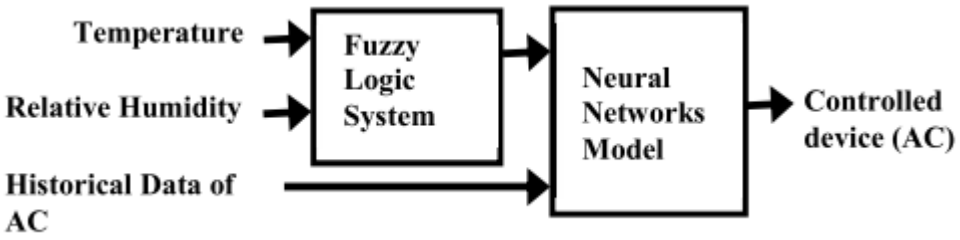


Figure 4 - Block Diagram of Air Condition Fuzzy-ANN Controlling Unit [24]

Above proposed system is a simplified system regarding what have been previously defined as a real SHEMS, but it is helpful to understand how smart optimization of SHEMS can be done based on intelligent tools; one first stage for classifying the data using genetic-fuzzy logic algorithm, and a second stage using neural networks is becoming a useful tool for energy management.

3. METHODOLOGY - DOMOTICS COST EFFECTIVE ELEMENTS

At this chapter it will be reviewed the technology basics that have been adapted to domotics to become it cots affordable. Here, the target is not the final devices, but the elements used on them.
Massive production of such elements is what has allowed price reduction on the devices; but final users cooperation for combining different applications and brilliant independent programmers working on their own and sharing with the community their achievements is what has develop th applications.

3.1 Sensors/Actuators boards

Commercial domotics devices have been develop based on modules using low cost and low power requirements chips which are connected to basic actuators and sensors. Those chips are:

3.1.1 ESP8266

Developed in 2014, ESP8266 is a system on a chip (SoC) microcontroller, low cost Wi-Fi microchip, with a full TCP/IP stack and microcontroller capability produced by Espressif Systems. Complete datasheet of the latest release of this chip is included on the Appendix.
Basic features of this chip are:

- CPU Tensilica L106 32 bits RISC processor.
- Operation voltage: 3-3.3 VDC.
- Operating current: Average value: 80 mA. Power saving architecture based on three basic modes of operation: active (170 mA), sleep (0.6 mA) and deep sleep (20 μA).
- Operating temperature range: -40 to 125 ºC. This provides high durability to it.
- Memory[1]: 32 kB instruction RAM, 32 kB instruction cache RAM, 80 kB user-data RAM, 16 kB ETS system-data RAM.
- External memory: Up to 16 MB is supported (depending on board used, 512 KiB-4 MiB are typically used). Note that chip doesn't have program memory, so external memory use is mandatory.
- IPv4 and TCP/UDP/HTTP/FTP protocols, Wi-Fi based (802.11 b/g/n). Though it is not natively available HTTPS protocol can be implement using a TLS1.2 client/server software.
- 17 general purpose input/output ports: but can only be used 9 or 10 of them. GPIO 16 is used for RTC; they can be configured using pull-up/down resistors. Also, they can be used as Pulse Width Modulation inputs.
- 10 bits Analog to Digital Converter (Tout pin, 0-1 VDC).
- Communication interface buses: SPI (Serial Peripherical Interface), I2C and UART (Universal Asynchronous Receiver Transmitter).

[1] Note that some documents refer those memory values in KiB and MiB: kibibyte, equivalent to 2^{10} bytes and 2^{20} bytes. But official data sheet form vendor shows values in kB (10^3 bytes) and MB (10^6 bytes).

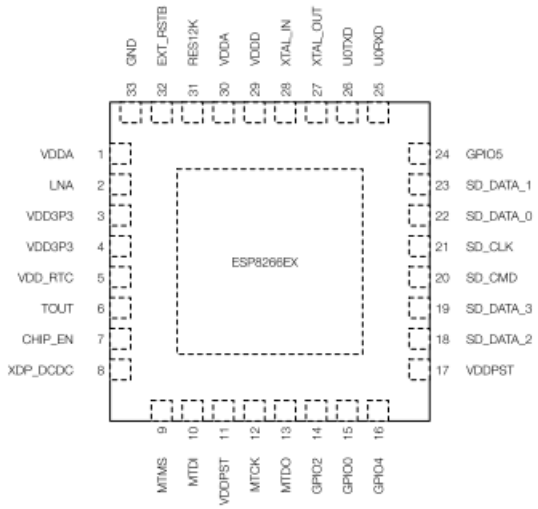


Figure 5 - ESP 8266 pin layout (top view) [25]

In the same way that Arduino is a print circuit board (PCB) using Atmel microchips, ESP 8266 is commonly used based on PCBs. Note that ESP 8266 has no program memory. Following list include just a few examples of what we can find on market.

- ESP-01: Develop by AI_Thinker company. With a reduced price (3 €), it makes available 2 GPIO pins for sensors and actuators control. There are two additional pins which can be used also as GPIO, but the main function of those pins is serial communication, to allow programming of ESP 01. Following picture shows basic components of ESP01:

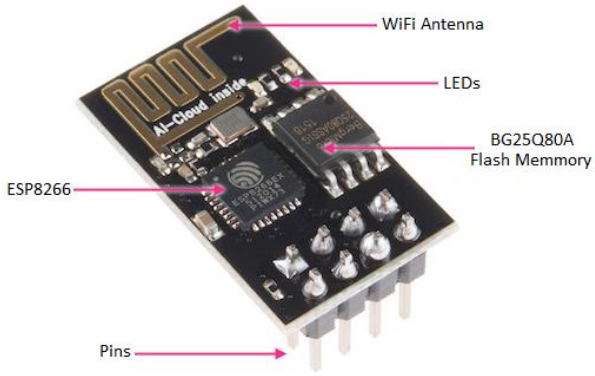


Figure 6 - ESP01 main components (back view) [26]

For ESP01 programming a USB/FTDI-TTL converter can be used, but also a Raspberry PI or Arudino board, keeping ESP01 in UART mode (GPIO=LOW, GPIO2=HIGH, considering they have a pull-up resistor so by default they are in HIGH); after reset, those GPIO can be used as input/output.

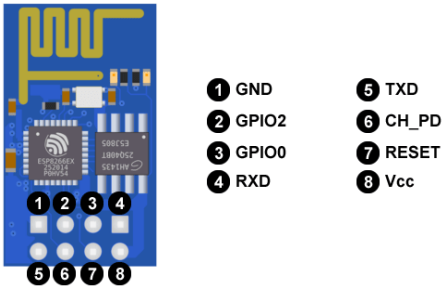


Figure 7 - ESP01 pins function [27]

- ESP05: It is the simplest version of board. It has been conceived for being a Wi-Fi Shield for Arduino. It has no GPIO ports available. It can be found around 4 €. It has a connector for an external antenna in order to increase Wi-Fi range.

Figure 8 - ESP05 upper view [28]

- ESP12: With a price around 4 € is the most used board nowadays. It is the most complete version, allowing access to the 11 ports GPIO, one of them is analog. It allows to be configured in sleep mode. It requires to be soldered in a mounting board (already included in the budged previously indicated).

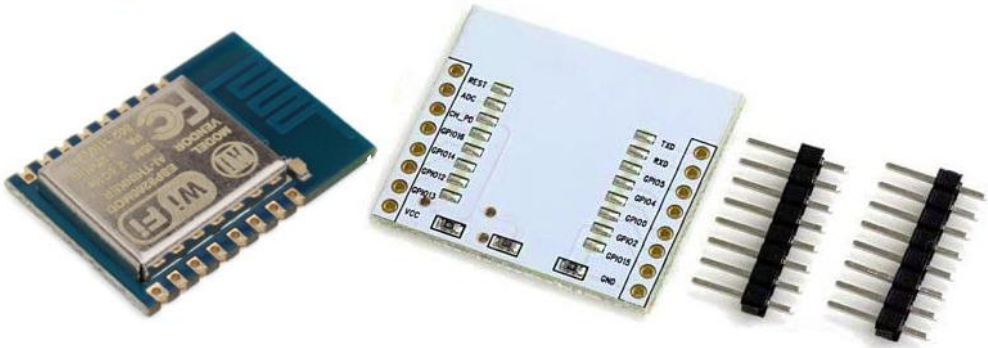


Figure 9 - ESP12 and adapter plate (mounting board), upper view [28]

- ESP201: it is a develop board. Price around 6 €. 11 GPIO ports can be accessed and it is designed to fit in a protoboard plate. Same as ESP05 it has a connector for an external antenna in order to increase Wi-Fi range.

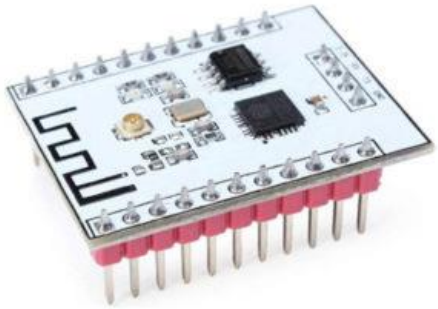


Figure 10 - ESP201 upper view [28]

- NodeMCU: Named after Node and Microcontroller (MCU). One board has a price around 8 €. Strictly speaking NodeMCU refers more to the firmware develop for ESP8266, more than a specific board. This firmware has been developed by ESP8266 open source community. It is based on ESP12 board, but boards use to come with mini-USB connector, already mounted serial/USB transducer, push buttons, … Regarding firmware, it uses Lua scripting language, through which Python, Basic or JavaScript languages can be used. This firmware enables programming through IDE Arduino.

Figure 11 - Typical board for using with NodeMCU firmware [28]

• SonOff Basic (4 €): SonOff basic is a complete device manufactured by Itead company, more than a board. It is a Wi-Fi switch, for using trough developer web server.

Figure 12 - SonOff Basic device [28]

What makes this device so interesting is that it is based on ESP8266, having the programming ports available in such way that it can downloaded a new firmware on it, so, e.g. it can enable MQTT communication (Tasmota, ESPurna, ESP Easy and other firmware), can be programmed in Python, Basic or JavaScript (NodeMCU firmware), ….
In SonOff board it is already implemented a power supply from 120-240 VAC, one relay commanded by GPIO12 and 10 A of interrupting current and wiring already done. So, at the end of the day, what we have is a compact Wi-Fi switch, programmable and customizable, ready to be integrated as a remotely controlled switch for 10 A loads, or connecting/disconnecting batteries, …
Referring to hardware, on the pictures bellow can be found the main components:

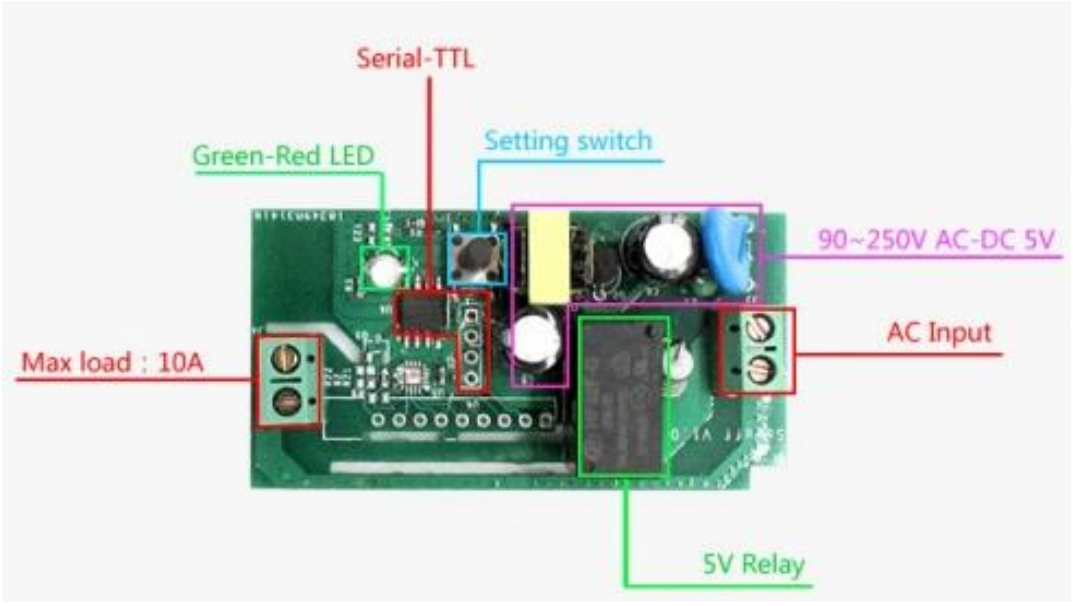


Figure 13 - SonOff Basic R1 main components and functions [29]

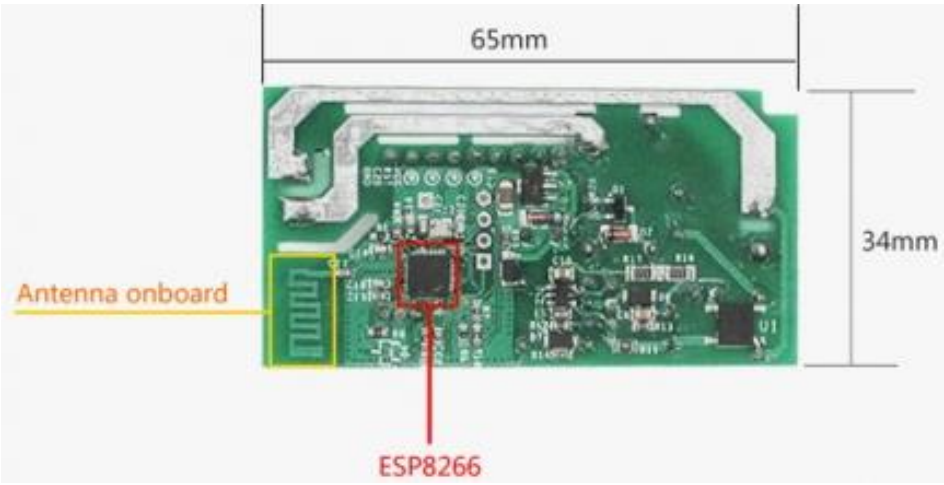


Figure 14 - SonOff Basic R1 dimensions [29]

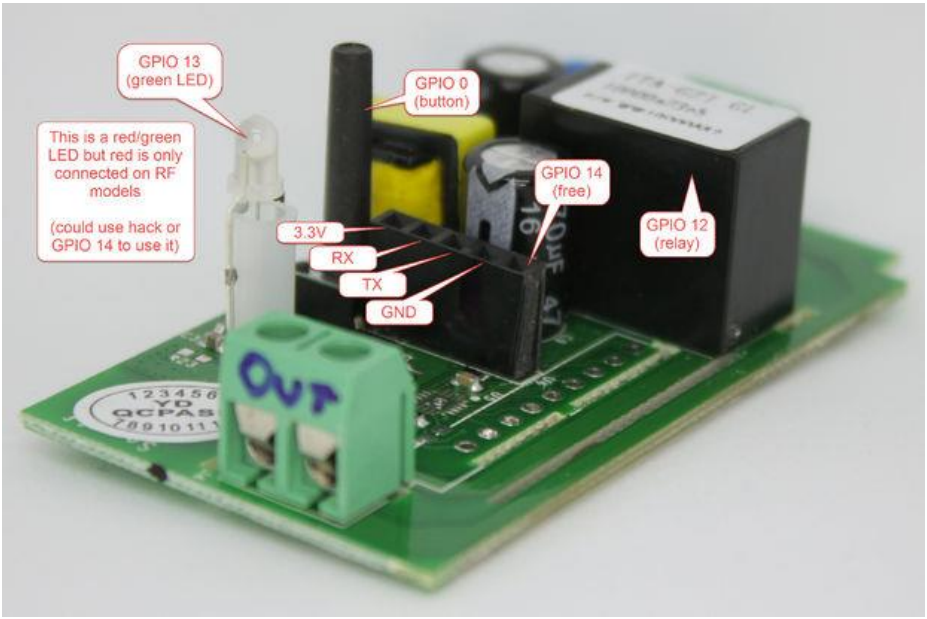


Figure 15 - SonOff Basic R1 pinout [29]

SonOff Basic schematic is provided in appendix.

- Another ITEAD/SonOff devices: based also in ESP8266, which allows to change the firmware to include the above referred features (MQTT, different programming languages, …) there are a bunch of devices. Follow a non-exclusive list of some of those devices, with a short description of functionality:
    - Switches: SonOff Basic, SonOff RF (with radio frequency added control), SonOff Dual (2 switches), SonOff 4CH (4 switches), SonOff 4CH PRO and PRO R2(4 switches and RF receiver), SonOff Touch (Gang Touch Wall

Switch), SonOff T1-T2-T3 (1-3 Gang Touch Wall Switch including RF control), Slamper & Slamper 2.0 (Bulb light switch with RF), SonOff S20 and S26 (outlet), SonOff S31 lite (outlet US standard), SonOff S55 (outlet for wall embedded mounting), 1CH Inching (relay module powered at 5VDC, dry contacts and 10 A burden).

- Switches with power measuring capability: SonOff POW, SonOff POW R2, SonOff S31 (outlet).
- Switches with Humidity and Temperature sensor: SonOff TH10, SonOff TH16.
- Dimmable E27 LED Lamp RGB Color Light Bulb: SonOff B1.
- RF-WiFi bridge: SonOff RF Bridge (Bridge between 433MHz RF and Wi-Fi)
- SonOff Zigbee devices: Taking advantage of the low power requirement of ZigBee there are some devices available: ZigBee Bridge (bridge between Wi-Fi and ZigBee), Wireless Switch SNZB-01, temperature and humidity sensor (SNZB-02), movement sensor (SNZB-03), Door-Window sensor (SNZB-04), …

3.1.2 ESP32

Develop in 2016 by Espressif Systems [30], ESP32 is the successor of ESP8266. It is a low cost, low power system on a chip microcontroller (SoC), with integrated Wi-Fi and dual mode Bluetooth [31]. Complete datasheet of the latest release of this chip is included on the Appendix.
Basic features of this chip are:

- CPU Tensilica Xtensa LX6 microprocessor single or dual core, 32 bits processor. 200 or 400 or 600 MIPS. ULP co-processor.
- Operation voltage: 2.3-3.6 VDC.
- Bluetooth: v4.2 BR/EDR and BLE (shares the radio with Wi-Fi).
- Operating current: Average value: 80 mA. Power saving architecture based on six basic modes of operation: active (240-100 mA depending on Wi-Fi, BLE mode), modem sleep (at 80 MHz, 20-31 mA for dual core or 20-25 mA for single core), light sleep (0.8 mA), deep sleep (150 μA ULP processor active, 100 μA ULP sensor monitored pattern, 10 μA RTC timer + RTC memory) and Hibernation (5 μA, just RTC memory) and power off mode (with CHIP-PU at low level, 1 μA).
- Operating temperature range: -40 to 125 ºC. This provides high durability to it. Models ESP32-D2WD and ESP32-U4WDH has integrated some chips with temperature range -40ºC to 105ºC, reducing their maximum temperature of system to 105ºC.
- Memory: 448 kB ROM for booting and core functions; 520 kB for data and instructions; 8 kB for SRAM in RTC fast memory accessible during boot and deep sleep mode; 8 kB for SRAM in RTC slow memory accessible during deep sleep mode by co-processor; 1000 bit of eFuse: 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including flash-encryption and chip-ID. Embedded memory depending on model 0-4 MB.
- External memory: Up to 16 MB is supported or SRAM 8 MB (no need for refreshing circuit).
- Models available: keeping in mind this paper is focused in low cost, remark that the price for different models of chips are going from 1,43 € up to 3.45 €:

- ESP32-D0WD, Dual core, no embedded memory, 5 x 5 mm.
    - ESP32-D0WDQ6, Dual core, no embedded memory, 6 x 6 mm.
    - ESP32-D2WD, Dual core, 2 MB embedded memory, 5 x 5 mm.
    - ESP32-S0WD, Single core, no embedded memory, 5 x 5 mm.
    - ESP32-U4WDH, Single core, 4 MB embedded memory, 5 x 5 mm.
    - ESP32-D0WD-V3, Dual core, no embedded memory, 5 x 5 mm. ECO V3 series.
    - ESP32-D0WDQ6-V3, Dual core, no embedded memory, 6 x 6 mm. ECO V3 series.

ECO V3 is a series which changes over the base modules consists in manufacturing them in one wafer-level. Additionally, some minor bugs have been corrected and as improvement, minimum baud rate in CAN module has been reduced from 25 kHz to 12.5 kHz.

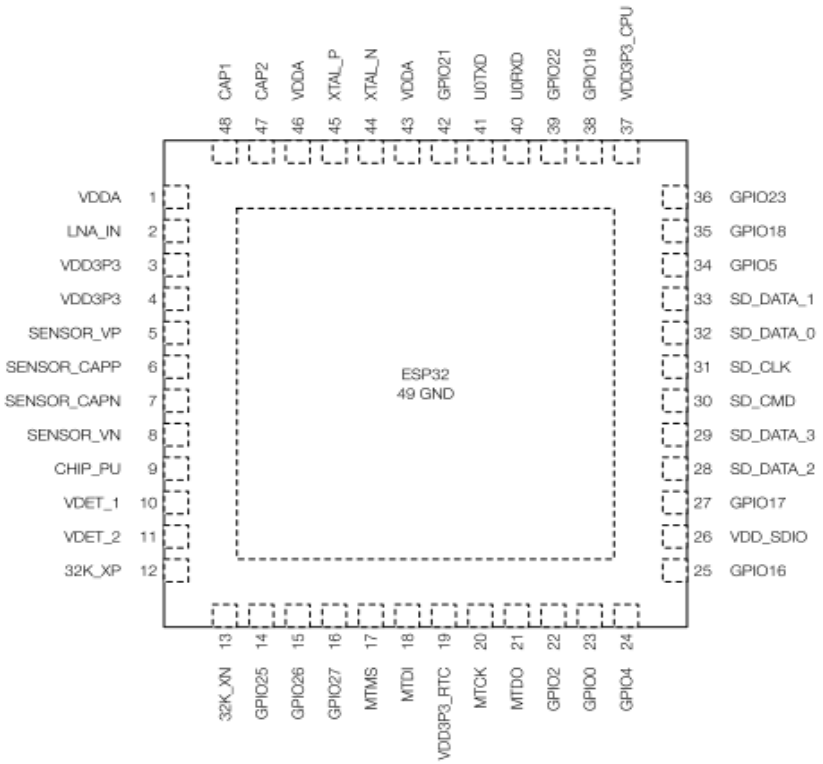


Figure 16 - ESP32 pin layout (QFN 6*6, top view) [32].

Another interesting pinout map for ESP32 can be seen on [33].

- FTP, HTTP, IBM MQTT, IPv4, IPv6, SSL, TCP, UDP protocols, Wi-Fi based (802.11 b/g/n). Manufacturer project HTTPS server provides a complete library for using HTTPS protocol [34].
- 34 general purpose input/output ports. They are used for different additional functions digital-only, analog-enabled, capacitive-touch-enabled, etc. Analog-enabled GPIOs and Capacitive-touch-enabled GPIOs can be configured as digital GPIOs but not the opposite. They can set pull-up, pull-down or high impedance in configuration.

- 2 analog inputs based on successive approximation analog to digital converter (ADC) based on 12 bits inputs. CPU can be configured to be wake-up depending on pre-defined threshold.
- 2 8 bits Digital to Analog outputs (DAC).
- Hall sensor: Chip integrates a Hall Sensor based on a N-carrier resistor. This resistor can be connected to an analog input, so magnetic field can be measured. Note that this sensor is not calibrated. It can be applied to implement a proximity sensor magnet based.
- Touch sensors: 10 capacitive-sensing GPIO are integrated to detect touch or approach of fingers.
- Pulse counter inputs.
- Pulse Width Modulation.
- LED PWM output (16 channels).
- Infrared remote controller: The infrared remote controller supports eight channels of infrared remote transmission and receiving.
- ULP: Allows to run micro-programs when CPU is sleeping.
- SD/SDIO/MMC Host controller: allowing reading/writing on storage systems such as SD, MMC, … cards.
- Communication interface buses: SPI (Serial Peripherical Interface), SPDIO (SDIO Card Specification v2), I2C, I2S and UART (Universal Asynchronous Receiver Transmitter).
- Hardware accelerator for general algorithms such as AES (FIPS PUB 197), SHA (FIPS PUB 180-4), RSA, and ECC. Those accelerators increase significantly the operation speed of microprocessor. They also support code encryption and dynamic encryption.

Quick guide for getting started with ESP32 can be found on [35].

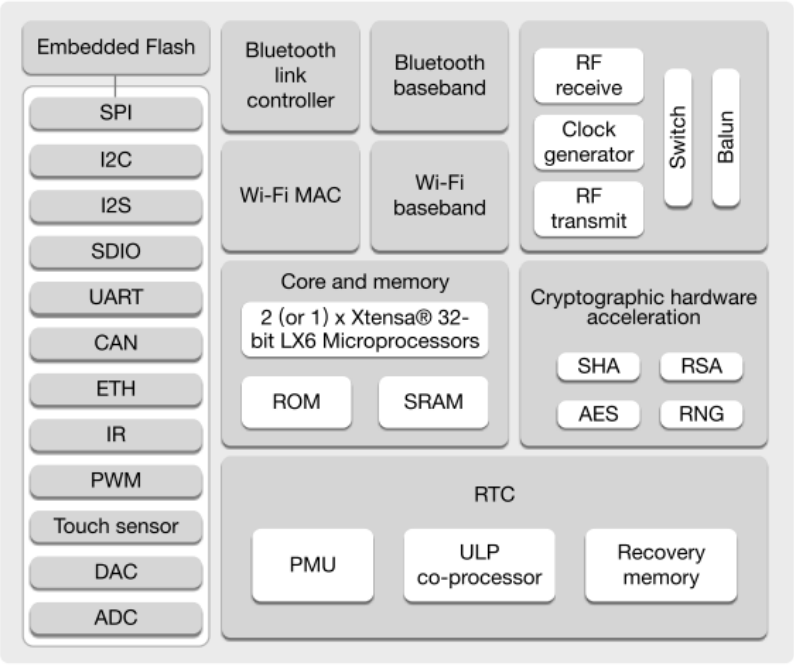


Figure 17 - ESP32 Function block diagram [36].

Same as ESP8266, ESP32 is available mounted in different boards, according to final purpose. This list are just a few examples of what we can find on market:

- ESP32 DEVKIT DOIT (5.59 €): There are two versions, 30 and 36 GPIOs, but the added pins available on 36 GPIOs version are connected to internal flash memory, so they are difficult to use. Pins are labelled on the board. Additionally, to chip ESP-WROOM-32 functions, on the board has been added the following features: reset and boot buttons, USB to UART interface (for easier programming) and voltage regulator circuit.

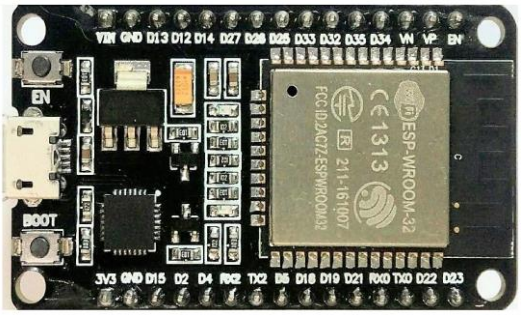


Figure 18 - ESP32 DEVKIT DOIT top view [28]

- Wemos Lolin32 (8,50 €): Similar to previous ones but adding an 0.96” OLED display. This allows basic interface with user.

Figure 19 - Wemos Lolin32 top view [28]

- ESP32 SX1278 (LoRa): This board comes with a SX1278 chip which is a LoRa transceiver chip. LoRa (Long Range) is a low-power wide-area network (LPWAN) protocol developed by Semtech. It is based on spread spectrum modulation techniques derived from chirp spread spectrum (CSS) technology. It was developed by Cycleo of Grenoble, France and acquired by Semtech, the founding member of the LoRa Alliance [37]. LoRa is intended for long range communications, small amounts of data and low power consumption. This board also includes 0.96” OLED display.

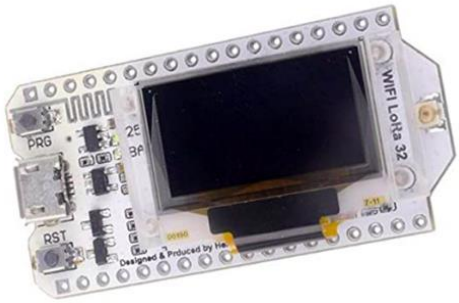


Figure 20 - ESP32 SX1278 top view [28]

- ESP32 CAM (6.99 €): This board includes 2 Mpx camera OV2640. Also includes TF support card up to 4G. Just 9 GPIO are available and it has no added the USB to UART interface. Some applications for this board are: surveillance camera, take photos and videos to SD card, PIR motion detection with Photo capture, take photo and display on Web server, …

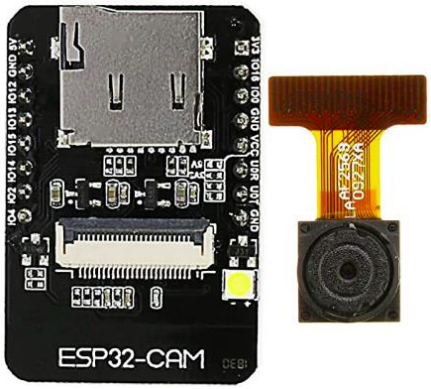


Figure 21 - ESP32 CAM top view [28]

- TTGO T-Call ESP32 (11.99 €): Main advantage of this board is to include SIM800L GSM/GPRS module, allowing connecting to the internet using your SIM card data plan, or communicate with the board via SMS or phone calls. Note that this module just allow connection to 2G network.

Figure 22 - TTGO T-Call ESP32 top view [28]

More details comparing different ESP32 boards models can be found on [38], [39], [40], [41] and [42].

3.1.3 Raspberry PI zero W

Develop by Raspberry foundation. Next chapter it will go in deep in other Raspberry products, but this model must be included in this section, though their technical specifications are above the simple sensor/actuator boards, price of 10.53 € allows to include it in the same group.
Launched to market on 2017, includes WiFi and Bluetooth LE which enables to include this model in this category (Raspberry "just" Zero (not W) has no Wi-Fi nor BLE so it is not useful as sensor/actuator).
System on a Chip, SoC for this model of Raspberry is Broadcom BCM2835 (Periphals specs. Included on appendix):

- CPU ARM1176JZFS, 700 MHz (1 GHz overclocking)
- GPU: Videocore 4: H.264 at 40 MBits/s, Fast 3D core access, 1Gpixel/s, 1.5Gtexel/s or 24 GFLOPs.
- FPU (Floating Point Unit): VFPv2

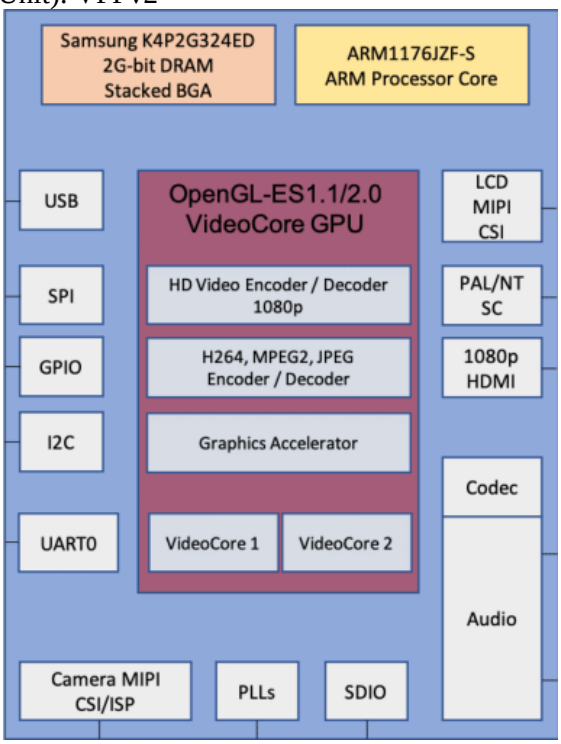


Figure 23 - Broadcom BCM2835 blocks diagram [43]

Main features of Raspberry Pi Zero W are:

- 802.11 b/g/n wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GHz, single-core CPU
- 512MB RAM

- Mini HDMI and USB On-The-Go ports
- 5 VDC Micro USB or GPIO power, 100 mA average when idle, 350 mA maximum with keyboard, mouse and monitor connected.
- Composite video and reset headers
- CSI camera connector
- HAT compatible 40 GPIO header (WH version has those GPIO already soldered to board).
    - 17 GPIO
    - UART
    - I2C
    - SPI
    - PCM (Pulse Code Modulation)
    - PWM
    - Chipset
- Micro SDHC slot (SD card will include OS)
- Composite video

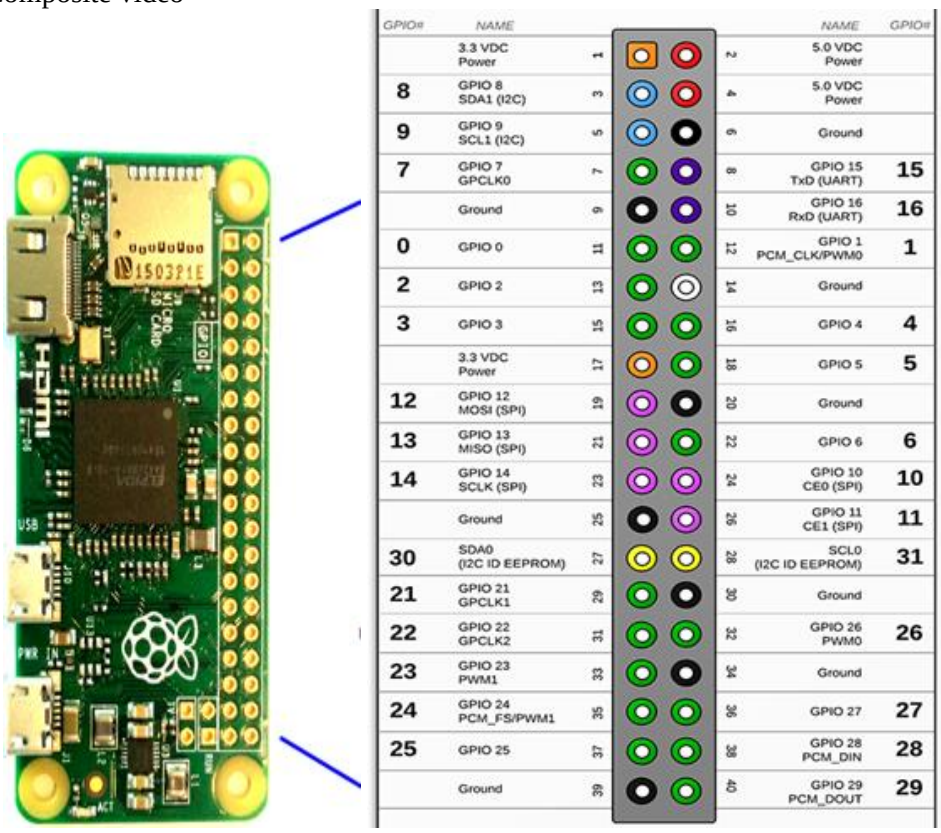


Figure 24 - Raspberry Pi Zero W pinout [44].

Operative system confirmed to be used in this board are NOOBS and Raspian (Jessie), LibreELEC, RetroPie, KaliLinux, MusicBox, MotionEyeOS, RuneAudio, ArchLinux, PuppyLinux, PwnPi, Open Media Vault. Other OS supported by Raspberry are to be confirmed.

GPIOs 0-53 and can work as input, output, or advanced function, though, not all GPIO can get all possible functions. Refer to BroadCom BCM2835 peripherals for detailed information. Available functions are: BSC master/slave (I2C), general purpose clock, SPI interface, PWM (2 inputs), UART interface, PCM Audio, secondary memory interface and chip interface. We don't find DAC or ADC or Hall Sensor, which makes this module less accurate as sensor/actuator, and there is no camera or relays integration and current consumed is considerably higher than other modules … but, in the other hand, provides a processing capability and high level programming possibilities that are not available in more dedicated modules.

3.2 Smart Centres Technologies

In order to develop the Smart Centres, there are available basic devices which can be assimilated to personal computers, but with a cost of just a little part of the cost for a personal computer. They will be the brain of our low cost SHEMS:

3.2.1. Raspberry PI:

This is the most extended and main option. The basic definition of this device is "a computer of the size of a credit card which can connect to a TV or monitor and a keyboard". Note that size is a little above a credit card: from 85.6x56.5 mm (most distributed models) to 65x56.5 mm (models 1A+ and 3A+) or 65x30 mm (Zero). The idea was born in 2006, but until 2012 was not launched to market. It has been developed by the Raspberry Pi Foundation to put the power of computing and digital making into the hands of people all over the world [45].
PI computers are used in many ways: as a Web server, media centre, robot and model railroad controller, YouTube live streamer, NAS storage, network monitor, digital photo frame, security system, game server, desktop computer, domotics control centre (of course) and whatever can be though … They give hobbyists and tinkerers an economical computer to experiment with, and although they were first touted as a learning tool, a Raspberry Pi can run myriad applications.
Existing models:

Raspberry Model	Cost	Year	System on a Chip Model	GPU	Memory
RPi1 A	25,00 €	2013	SoC Broadcom BCM2835 (ARM v6Z, 32 bits, CPU 1× ARM1176JZF-S 700 MHz, FPU VFPv2)	GPU Broadcom VideoCore IV @ 250 MHz	256 MB SDRAM
RPi1 B	35,00 €	2012	SoC Broadcom BCM2835 (ARM v6Z, 32 bits, CPU 1× ARM1176JZF-S 700 MHz, FPU VFPv2)	GPU Broadcom VideoCore IV @ 250 MHz	512 MB SDRAM
RPi1 A+	20,00 €	2014	SoC Broadcom BCM2835 (ARM v6Z, 32 bits, CPU 1× ARM1176JZF-S 700 MHz, FPU VFPv2)	GPU Broadcom VideoCore IV @ 250 MHz	512 MB SDRAM

RPi1 B+	25,00 €	2014	SoC Broadcom BCM2835 (ARM v6Z, 32 bits, CPU 1× ARM1176JZF-S 700 MHz, FPU VFPv2)	GPU Broadcom VideoCore IV @ 250 MHz	512 MB SDRAM
RPi2 B	35,00 €	2015	SoC Broadcom BCM2836 (ARM v7A, 32 bits, CPU 4× Cortex-A7 900 MHz, FPU VFPv3 + NEON)	GPU Broadcom VideoCore IV @ 250 MHz	1 GB SDRAM
RPi2 B v1.2	35,00 €	2016	SoC Broadcom BCM2837 (ARM v8A, 32 bits, CPU 4× Cortex-A53 900 MHz, FPU VFPv4 + NEON)	GPU Broadcom VideoCore IV @ 250 MHz	1 GB SDRAM
RPi3 B	35,00 €	2016	SoC Broadcom BCM2837 (ARM v8-A, 64/32 bits, CPU 4× Cortex-A53 1.2 GHz, FPU VFPv4 + NEON)	GPU Broadcom VideoCore IV @ 250 MHz	1 GB SDRAM
RPi3 B+	35,00 €	2018	SoC Broadcom BCM2837B0 (ARM v8-A, 64/32 bits, CPU 4× Cortex-A53 1.4 GHz, FPU VFPv4 + NEON)	GPU Broadcom VideoCore IV @ 250 MHz	1 GB SDRAM
RPi3 A+	25,00 €	2018	SoC Broadcom BCM2837B0 (ARM v8, 64 bits, CPU 4× Cortex-A53 1.4 GHz, FPU VFPv4 + NEON)	GPU Broadcom VideoCore IV @ 250 MHz	512 MB SDRAM
RPi4 B	35/55/75 € (depends on memory size)	2018	SoC Broadcom BCM2711 (ARM v8-A, 64/32 bits, CPU 4× Cortex-A72 1.5 GHz, FPU VFPv4 + NEON)	GPU Broadcom VideoCore VI @ 500 MHz	1 or 2 or 4 or 8 GB SDRAM

Table 1: Raspberry PI models (cost, year, SoC, GPU and memory) [46]

Raspberry Model	Interfaces	GPIOs	Power Ratings
RPi1 A	1 USB2.0, 1 camera interface (Camera Serial Interface, CSI), 1 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, SD, MMC, SDIO card slot	8 GPIOs plus UART, I2C, SPI	300 mA
RPi1 B	2 USB2.0, 1 camera interface (Camera Serial Interface, CSI), 1 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, SD-MMC-SDIO card slot. Ethernet 10/100Mbits/s	8 GPIOs plus UART, I2C, SPI	700 mA
RPi1 A+	1 USB2.0, 1 camera interface (Camera Serial Interface, CSI), 1 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, microSDHC card slot	17 GPIOs plus UART, I2C, SPI, HAT ID Bus	200 mA
RPi1 B+	4 USB2.0, 1 camera interface (Camera Serial Interface, CSI), 1 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, microSDHC card slot. Ethernet 10/100Mbits/s	17 GPIOs plus UART, I2C, SPI, HAT ID Bus	200-350 mA
RPi2 B	4 USB2.0, 1 camera interface (Camera Serial Interface, CSI), 1 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, microSDHC card slot. Ethernet 10/100Mbits/s	17 GPIOs plus UART, I2C, SPI, HAT ID Bus	220-820 mA
RPi2 B v1.2	4 USB2.0, 1 camera interface (Camera Serial Interface, CSI), 1 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, microSDHC card slot. Ethernet 10/100Mbits/s	17 GPIOs plus UART, I2C, SPI, HAT ID Bus	220-820 mA
RPi3 B	4 USB2.0, 1 camera interface (Camera Serial Interface, CSI), 1 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, microSDHC card slot.Wi-Fi (b/g/n/ac dual band 2.4/5 GHz), Bluetooth 4.1 BLE. Ethernet 10/100Mbits/s	17 GPIOs plus UART, I2C, SPI, HAT ID Bus	300 mA – 1.4 A
RPi3 B+	4 USB2.0, 1 camera interface (Camera Serial Interface, CSI), 1 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, microSDHC card slot, USB Boot mode. Wi-Fi (b/g/n/ac dual band 2.4/5 GHz), Bluetooth 4.2 BLE. Ethernet 10/100/1000 Mbit/s	17 GPIOs plus UART, I2C, SPI, HAT ID Bus	459 mA-1.13 A

RPi3 A+	1 USB2.0, 1 camera interface (Camera Serial Interface, CSI), 1 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, microSDHC card slot. Wi-Fi (b/g/n/ac dual band 2.4/5 GHz), Bluetooth 4.2 BLE	17 GPIOs plus UART, I2C, SPI, HAT ID Bus	--
RPi4 B	2 USB2.0, 2 USB 3.0, 1 camera interface (Camera Serial Interface, CSI), 2 HDMI port, RCA jack composite video, Display Serial Interface (DSI) used for LCD and similar displays, audio input via I2C, audio output via jack, microSDHC card slot, USB Boot mode, Wi-Fi (b/g/n/ac dual band 2.4/5 GHz), Bluetooth 5.0 BLE. Ethernet 10/100/1000 Mbit/s.	17 GPIOs plus 4xUART, 4xI2C, 4xSPI, HAT ID Bus	600 mA-1.25 APower Supply USB-C or GPIO

Table 2: Raspberry PI models (interfaces, GPIOs, power) [46]

We can also find Raspberry Pi Zero, which has been commented in section 3.1.1 and Raspberry Pi Compute module, which being for industrial applications is out of scope of this work.
Operative System: OS is stored in the SD card. Though the vendor remains control of hardware platform, OS it is open source officially is an adapted version of Linux Debian distribution, named as Raspbian; but there are many different OS available such as Windows 10 IoT Core (former Windows Embedded), RISC OS Pi, Broadcom VCOS, Free BSD, Open BSD and NetBSD (Unix like OS), Plan 9 from Bell Labs and Inferno (also UNIX like, from Bell Lab), Haiku, HelenOS, and other Linux based: Ubuntu, LibreELEC (short for Libre Embedded Linux Entertainment Center), Android Things (Android for IoT), Fedora, … and 21 more Linux based OS. It supports Python and Scratch as main programming languages.

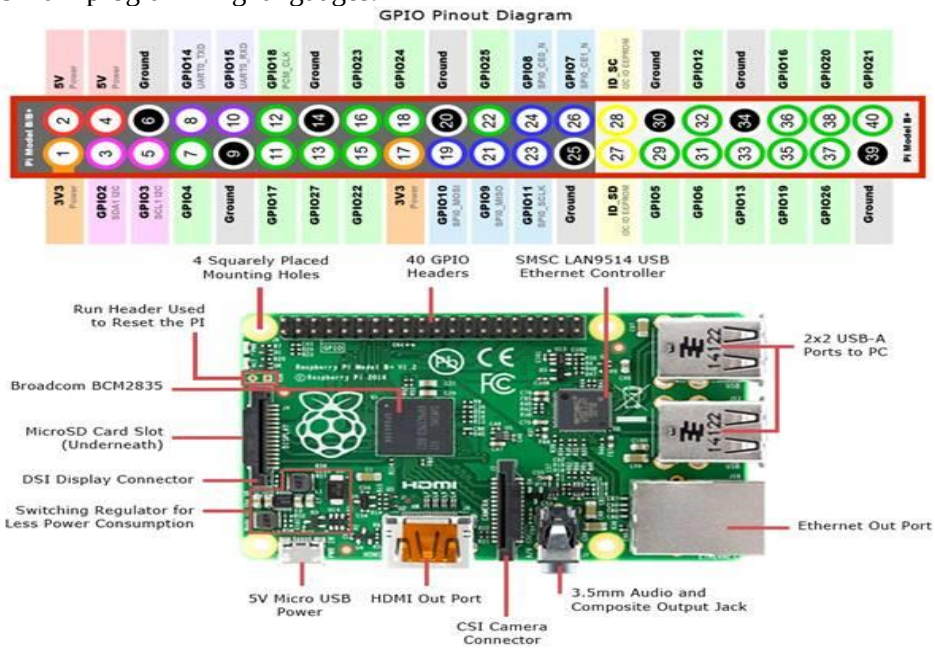


Figure 25 - Hardwarre specifications for Raspberry Pi (model 2B) [47].

3.2.2. Cubieboard:

Launched to market on 2012. It can work with Android 4 ICS, Ubuntu 12.04 desktop, Fedora 19 ARM Remix desktop, Armbian, Arch Linux ARM, a Debian-based Cubian distribution, FreeBSD, or OpenBSD. From 2012 there has been many actualizations: Cubieboard1 (49.84 €), Cubieboard2 (49.92 €), Cubietruck (83.10 €), Cubieboard4 (107.67 €), Cubieboard5 (89.81 €). Let's see specifications for Cubieboard 2 and CubieBoard5, since they are the most representative models:

CubieBoard2:

- SoC: AllWinner A20
  - CPU: Dual core ARM Cortex-A7 MPCore, 1.5 GHz
  - GPU Dual Core Mali-400MP2
  - video acceleration: CedarX able to decode 2160p video
- 512 MiB (beta) or 1GiB (final) DDR3
- 4 GB NAND flash built-in, 1x microSD slot, 1x SATA port.
- 10/100 Ethernet connector
- 2x USB Host, 1x USB OTG, 1x CIR.
- 96 extend pin including I²C, SPI, LVDS
- Dimensions: 10 cm × 6 cm

Cubieboard5:

- SoC: Allwinner H8
  - CPU: ARM Cortex-A7 @ 2 GHz octa-core
  - GPU: PowerVR SGX544 @ 700 MHz
  - display controller: Toshiba TC358777XBG, supports HDMI 1.4 1080p and DisplayPort, no LVDS support
- 2 GiB DDR3
- 8 GB EMMC flash built-in, 1x microSD slot, 1x SATA 2.0 port (Hard Disk of 2,5") via USB bridge.
- 10/100/1000Mbps RJ45 Gigabit Ethernet
- 2x USB Host, 1x USB OTG, 1x CIR.
- S/PDIF, headphone, and HDMI audio out, mic and line-in via 3.5mm jack, and onboard mic.
- Wi-Fi (dual-radio 2.4 and 5 GHz) and Bluetooth on board with PCB antenna
- 70 extended pins including I²C, SPI
- Dimensions: 11 cm × 8 cm

3.2.3. Gooseberry PI:

Cost a little above 40 €. A10 ARM Cortex-A8 processor, Mali 400 graphics, 512 RAM memory, 4 GiB storage, Wi-Fi, SD Card input, mini-HDMI port, USB port and Android OS by default.

3.3 Firmware and Software

Even when low cost hardware has been one key factor for low cost domotics development, the actual strongest factor has been community firmware and software open source develop. There is a vast and huge network of developers working in open code but also supporting to users, and

sharing knowledge through public channels, forums and dedicated webs (a list of such knowledge channels is provided on appendix).

3.3.1. ESP 8266 based devices firmware

One most used base board for domotics devices is ESP8266 and this is mainly due to different firmware versions for this SoC (System on a Chip). Most of them has been extrapolated to ESP32. Following are the main options for firmware:

3.3.1.1. ESPEasy

Launched in 2015, was the first alternative firmware. Develop by Letscontrolit community its source code is available at [48]. The firmware is built on the ESP8266 core for Arduino which in turn uses many open source projects. Same than other alternatives, it is a web configurable multisensor (with some actuator things in progress). It started with generic ESP8266 platform and added documentation on how it can be used for fabricated boards, such as Sonoff. This approach makes it very flexible and able to support any combination of sensors / actuators but requires a bit more configuration to get started.
Project is updated many times a day, mostly by few core developers from Lets Control It, but they also merge code from other contributors, which is quite nice and in the spirit of true OpenSource project.community size is big, with a balance between developers and "regular" users.
Once firmware is downloaded on the device using UART port (TTL levels, Rx-Tx, it can be used a USB/TTL converter, an Arduino board, a Raspberry Pi device, …), and using SonOff button for entering ESP8266 in programming mode, and once device is restarted a Wi-Fi network owned by device can be accessed.
We can connect to Wi-Fi network "esp_0", and at http://192.168.4.1 it can be found the embedded webpage for configuring home Wi-Fi network and password.
After rebooting the device will connect to home Wi-Fi network, obtaining an IP address from home router DHCP server. Connecting to such new IP address in our web browser we can start to configure it and manage our device.

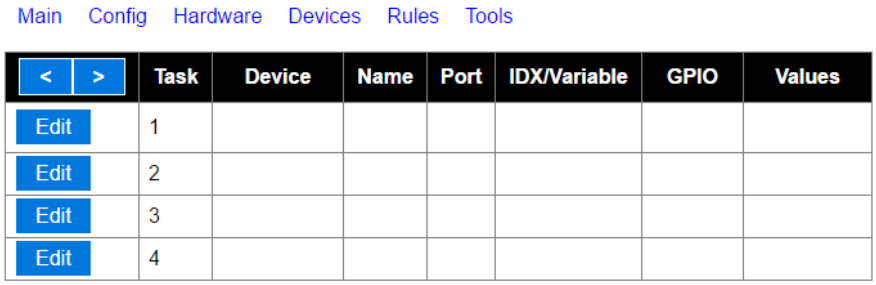


Figure 26 – Tasks table page on ESPEasy [49].

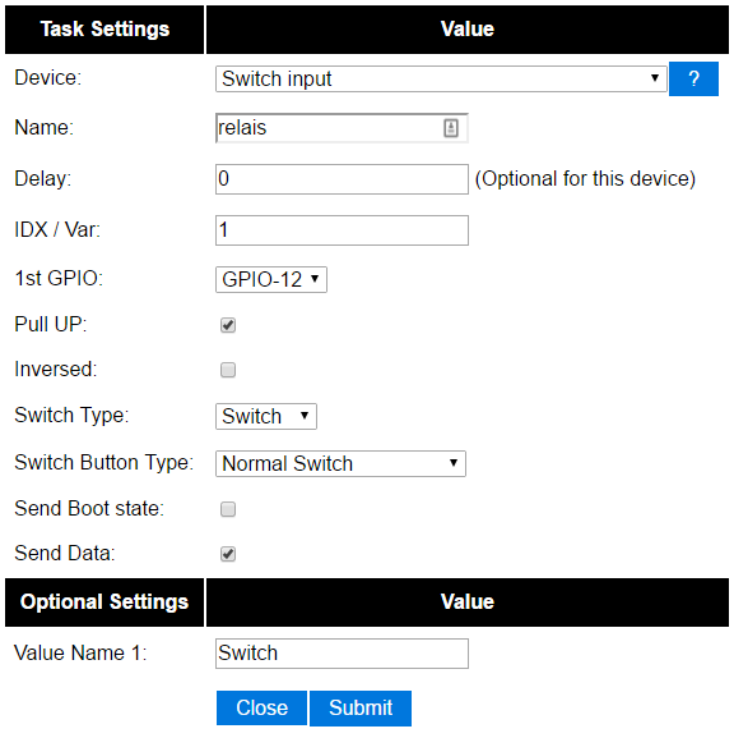


Figure 27 - Configuring page ESPEasy allowing to configure relay module [49].

After rebooting the device will connect to home Wi-Fi network, obtaining an IP address from home router DHCP server. Connecting to such new IP address in our web browser we can start to configure it and manage our device.
Scripts can be used, of the type:

On TurnOn do

gpio,12,1

EndOn

On TurnOff do

gpio,12,0

EndOn

The http command to switch the Sonoff to On will be: *http://<ip-of-Sonoff>/control?cmd=event,TurnOn*
The http command to switch the Sonoff to Off will be: *http://<ip-of-Sonoff>/control?cmd=event,TurnOff*
It supports:

- Various WiFi connection modes, including AP (hotspot) or STA (client) mode, with multiple SSID configuration and Wifi Network scanning.
- Over-the-Air update (OTA) and configuration backup and restore.
- Logging over some interfaces, such as Serial, Syslog or HTTP, with configurable log levels.
- Time synchronization via NTP

In order to use with SHEMS Center (applications) it is important considering protocols available for the device [50]:

- Domoticz HTTP
- Domoticz MQTT
- OpenHAB MQTT
- PiDome MQTT
- Nodo Telnet
- ThingSpeak
- EmonCMS
- Generic HTTP
- Generic HTTP Advanced
- InfluxDB HTTP Api (via Generic HTTP Advanced)
- Nettemp HTTP Api (via Generic HTTP Advanced)
- FHEM HTTP
- Generic UDP

3.3.1.2. SonOff Tasmota

Project started in 2016 (but officially launched in 2017). The name stands for: Theo-Arends-Sonoff-MQTT-OTA, being Theo Arends the developer. Supports many more ESP8266 based boards, much more than mere Son Off [51]. Project is practically daily updated, by Theo, reworking some of the contributions made by community. This community is a mix of developers and users, a little unbalanced, in reference to ESPEasy community; community produce regularly Youtube tutorials and wiki pages, being important part of the success of this firmware.
Most of the features are the same than ESPEasy.
One significant difference is that Tasmota has all possible devices integrated in just one code: this makes it easy to use but produces a heavier code.
It has same protocols than ESPEasy, but they have not been customized per applications, so, instead of MQTT Domoticz, what we have is generic MQTT and some parameters which must be adapted to Domoticz application. In the other hand, some IR protocols are available, allows mDNS MQTT discovering mode and 433 MHz receiving, learning and sending codes.
The programming method is the same than for ESPEasy and it will be the same than for next analysed firmware, ESPurna. Same that other firmware it can be downloaded using a Raspberry or even Arduino IDE, as it shown in [52]
Once the device has been flashed, the Wi-Fi connection available is tasmota_XXXXXX-####, now the IP address is http://192.168.4.1.

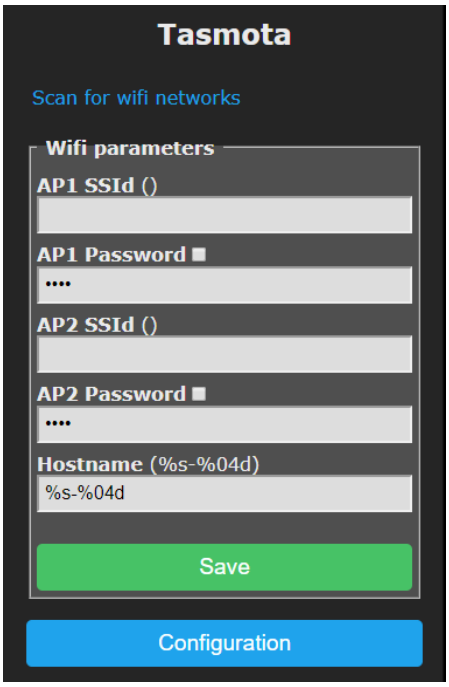


Figure 28 - Configuring Wi-Fi connection for Tasmota [53].

As it can be seen on above picture, two different Wi-Fi networks can be configured for our device. Now, MQTT must be configured.

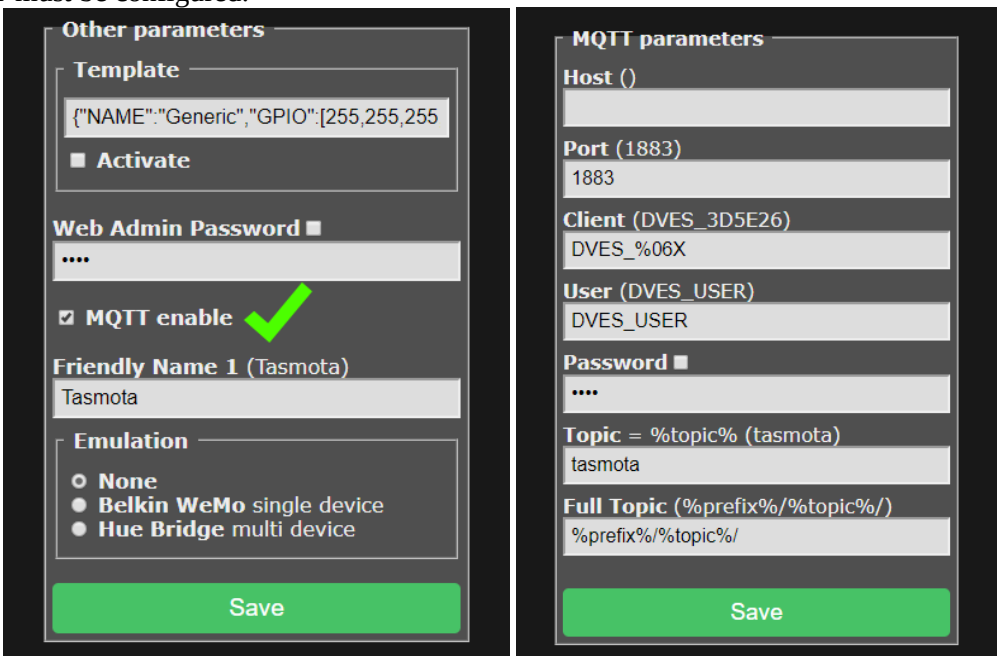


Figure 29 - Configuring MQTT protocol in Tasmota [53].

Next step will be to configure template or module, selecting in such way the device that we are using (as previously said, firmware is the same for all different devices).

3.3.1.3. ESPurna

ESPurna: Latest project to be added as firmware option for ESP8266 boards, developed by Xose Perez 2016. The name stands from Catalonian word "espurna" which means "spark" in Catalan [54]. Supports many boards [55]: practically all ITEAD SonOff, Lights/LED controllers (AI Thinker, Arilux, iWoole, LOHAS, Deltaco, Xiaomi), power plugs (Xenon, HAMA, OBI, Xenon, Maxio, Teckin, Zhilde,, Blitzwolf, Kogan Smarter Home, Smartlife, …), development boards, Shelly, and custom and development boards. There is a community (smaller than Tasmota) of skilled developers but few regular users; not so much tutorials or YouTube videos. Page is updated almost daily directly from Xose Perez, many of them based on discussions with the community. Support documentation is the less of three options, but it is the most structured of them and easier to use.
Most of the features are the same than previous ones.
Remarkable features:

- Home Assistant (one of main Center SHEMS application) auto-discovery function making integration of ESPurna based devices automatically.
- Telnet support, if has been previously enabled in UI.
- 433MHz RF receiving / learning / sending 433 MHz codes
- mDNS, NetBIOS and LLMNR and SSDP advertising and discovery - mDNS advertising being especially interesting. Like Tasmota, ESPurna supports MQTT discovery, but with no failover.
- Direct Influx DB Integration, allowing to export metrics directly to InfluxDB, which is an open source application for monitoring metrics, events, … (to be installed on control center SHEMS).
- Tailored binaries and integration with NoFUSS update framework: NoFUSS is a web server implemented by Xose Perez in order to keep always updated devices firmware.
- The board that is going to be used must be defined before download the image, this means the firmware is specific for each device, making size of code accurate to device.
- And same than Tasmota, ESPurna just allow one MQTT server defined. This MQTT function allows automatic discover of MQTT servers, but it doesn't allow MQTT server commutation (failover system). Anyway, MQTT server works better than others.

It has same protocols than previous options, remarkable MQTT SSL/TLS. And it can be integrated with Thingspeak (HTPP API), Google assistant (IFTTT), Domoticz (MQTT), Alexa, Home Assistant, InfluxDB, …
Once the device has been flashed, the Wi-Fi connection available is ESPURNA-****** ("fibonacci" password), now the IP address is http://192.168.4.1 (user "admin", password "fibonacci").
Next step will be change password:

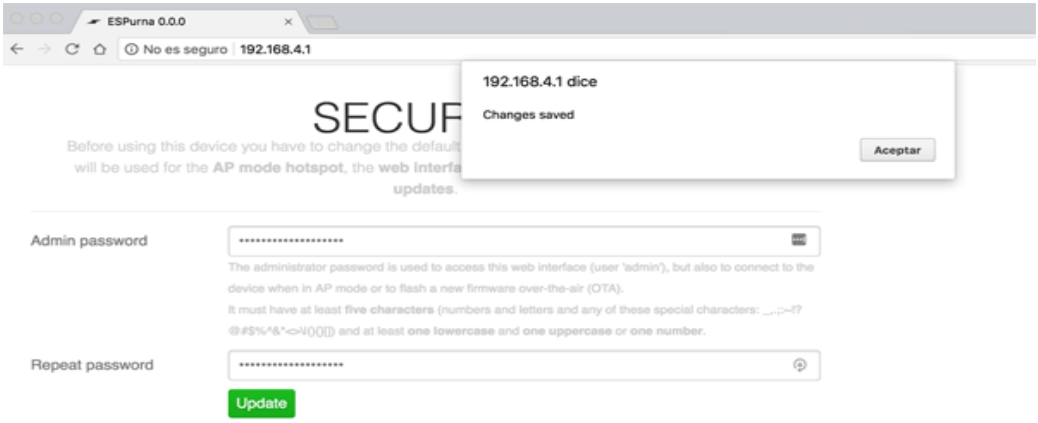


Figure 30 - Changing password in ESPurna [56].

Now, we access again, to the same Wi-Fi and then IP address, but with new password.

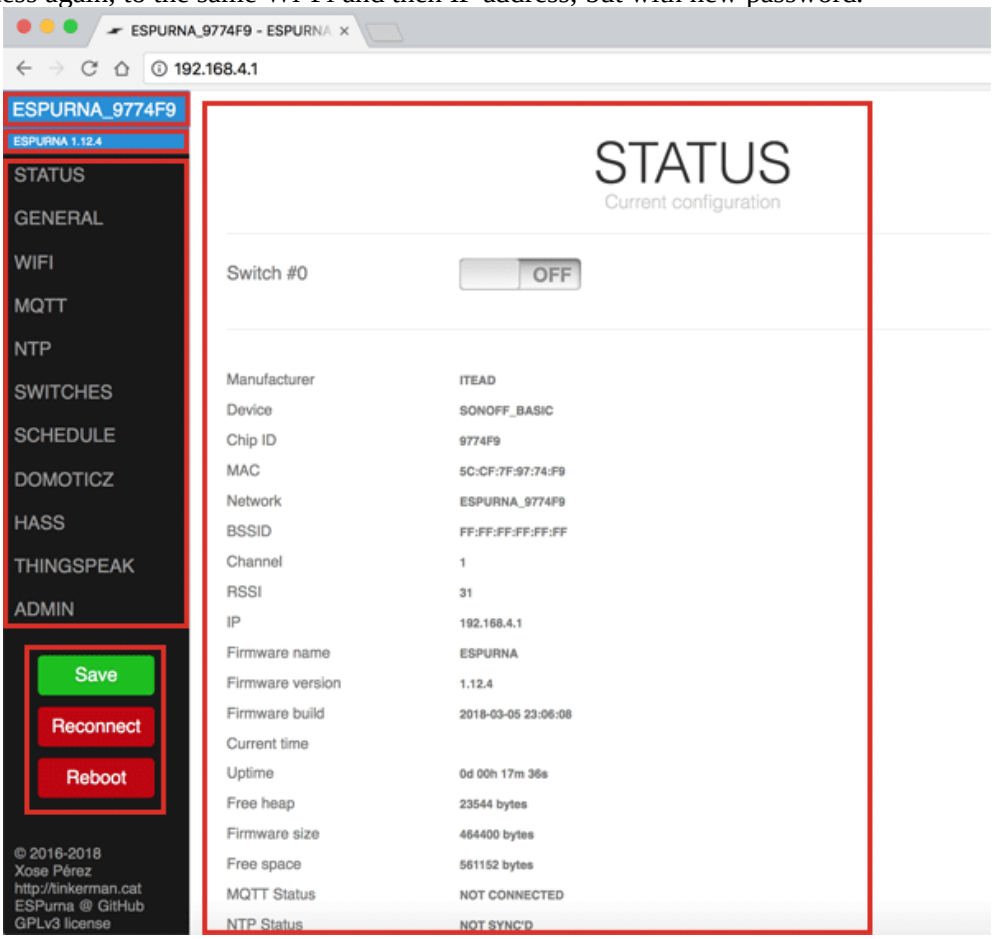


Figure 31 - General view UI for ESPurna[56].

There is no place in this paper for detailed explanation about all possibilities and configuration for this firmware. Let's provide just an overview. Above picture is showing main functions on UI:

- Upper left corner shows the name for this access point (AP).
- Just bellow it is showed firmware version.
- Bellow menu with different configuring options.
    - GENERAL: Host name, double click delay, LED mode (when device LED will light on), Alexa integration.
    - WiFi: It allows to configure one or more networks to which the device will connect for our home system (password, SSID, IP, Gateway, Mask, DNS).
    - MQTT (client): It must be already working the MQTT server before configuring this option. ON/OFF, MQTT broker IP, MQTT broker port, MQTT user (it can be blank), MQTT password (it can be blank), MQTT client (use to be the same than device host name), MQTT QoS (Quality of Service, fire and forget0-Fire and forget most once (0), at least once (1), exactly once (2), MQTT retain (client will receive the latest message of topic because server will keep it until it is delivered), keep alive (timing between keep alive messages), root topic (topic name), JSON payload (uses JSON format for messages).
    - NTP: Server address, time zone and DST enabling (Daily Saving Time).
    - Schedule: For programming ON/OFF periods.
    - Domoticz (Domoticz IN Topic, Domoticz OUT Topic, IDX -index), Home Assistant (discover, prefix, configuration), ThnigSpeak.
    - Depending on device, some other options can be showed, such as SENSORS or SWITCHES.
- Bellow Save/reconnect/reboot actions.
- On center/right information about this specific device. This will depend on the device we are using, including also a "swtich" for turn on/off the device.

The three firmware options are being developing for ESP32 now. They are on Beta version, having many of functions already implemented, but other ones still working on them.
All firmware lacks Secure communication /TLS, but for Tasmota and ESPurna which support TLS for MQTT, while web UI and other interfaces remains unencrypted and unauthenticated. This point makes cybersecurity a weak point in those systems … perhaps, considering the damage that can be done this is not a big issue, but it must be always kept in mind that devices controlled by those boards/firmware can be accessed remotely (a risk assessment must be done before installing or add a new device to the system).
Other common improvement area is to adapt them to IPv6 format and structured documentation; there is a huge amount of information on the web, looking for, almost everything can be solved … but it is required to look for it, because of not structured documentation.
We can also find domotics customized software for ESP32, as it is shown on [57], [58], [59], Tasmota adaptation for ESP32 on Beta release [60], ESPeasy adaptation for ESP32 on Beta release [61] or ESPHome especific domotics firmware for ESP32 [62] - [63].
For further details comparing ESPeasy, Tasmota and ESPurna can be found on [64] and [65]

3.3.2. SHEMS APPLICATIONS

They are the real "brain" of SHEMS, and they are the part which allows smart controls and different functions and algorithms for controlling. All of them can be implemented in any SHEMS

Smart Center, though we will focus in Raspberry PI implementation also they will work PCs or any other device.
All analysed options are open-source Internet of Things (IoT) application and API (Application Programming Interface) to store and retrieve data from things using the HTTP and MQTT protocol over the Internet or via a Local Area Network. There are many options out there, but we will focus on the three biggest communities: OpenHAB (2013), Home Assistant (2013) and Domoticz (2012), a big community translates into a big support; we will add Thingspeak (2010), one really interesting option due to its link with Matlab, the mathematical analysis tool for excellence. More about MQTT can be found on [66]

3.3.2.1. OpenHAB

Open Home Automation Bus, written in Java by Eclipse Foundation, based on the premise of connecting to devices and services from different vendors. It has one of the biggest communities.
Installation requires Java Virtual Machine but using the application OpenHabian the process is fast and automatic. Once installed just entering in the web UI devices (things) can be added.
Configuration is done using command lines editing files; but since 2017, with OpenHAB2 which integrates Paper UI most of functionalities can be done in a graphic environment … MOST, not all, so, at the end of the day, still it is required to edit files.
Flexibility is the core of this system, which also has a counterpart making it quite complicated to program; as commented, to enable all power of this system it is still required to edit files, which can be a tedious task since flexibility means also having to include a lot of parameters to define each option.
Pace of development is slow, which also means that when something is approved it has been well tested; but it will be not the first platform to include new devices and features.
Automation in this system allows integration of almost everything; but needing to edit files using Xbase programming language (Xbase derivates from dBase) is not the easiest way to work; using JSR223 plugging we will be able to program using JavaScript or Jython.
OpenHAB includes iOS and Android applications and tools for creating customized ones. It also works on the own hardware, not requiring any external connection (which adds stability to system), but it can be integrated with most popular applications (Google Assistant, Alexa, IFTTT, Apple Homekit, …) and also provides free allocation service on myopenhab.org [67].
OpenHAB is difficult to work with, but stable and flexible.

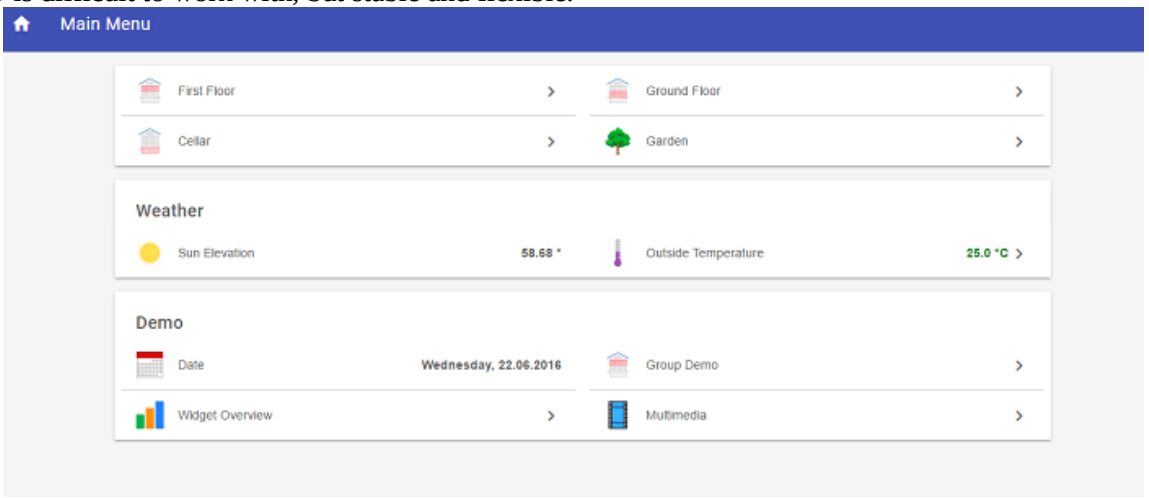


Figure 32 - OpeHAB UI [68].

3.3.2.2. Home Assistant

Written in Python 3 and founded by Paulus Schoutsen, it can be run on almost any platform and even can be mounted on a container for Docker system. Hass.io is a Raspberry Pi smart home OS that handles everything from installation of Home Assistant to managing the Home Assistant user interface (UI). There are loads of add-ons for Hass.io from Let's Encrypt to Alexa and Google Assistant. Therefore, Home Assistant for the Raspberry Pi delivers loads of functionality. The open-source home automation platform boasts a bevy of integrations including Plex, IFTTT, Sonos, Zigbee, Z-Wave, and Samsung SmartThings. Community of users for Home Assistant is the biggest and active of the different options.
As Paulus state on [69], the bases of Home Assistant are:

- YOU SHOULD NOT HAVE TO ADAPT TO TECHNOLOGY.
- YOU ARE NOT THE ONLY USER OF YOUR HOME AUTOMATION.
- LIMIT THE IMPACT OF FALSE POSITIVES AND NEGATIVES.
- THE PERFECT APP IS NO APP.
- YOUR SYSTEM SHOULD RUN AT HOME, NOT IN THE CLOUD.

Installation is similar to OpenHAB. Download and burn an image of HassBian (supported by Home Assistant project).
Configuration is done in the start-up self-discovering the "things" (devices) connected to home network. The routine can be repeated when required. If it is needed to customize installation it will require to edit YAML files.
Flexibility is not one of the main advantages of this system … at least if we use self-discovering function; but it is something we can expect: system tries to "figure-out" our requirements. But if we have "especial" requirements we will have to use YAML files.
Pace of development is fast since there is no detailed review and in the other hand, there is a vast community, which also means un-structured development and bugs.
Automation in this system is done based on YAML files; it can be classified as a pain on the neck … but Home Assistant requires to get familiar with Yaml; in the other hand, it is not so difficult, and we also have the help of AppDaemon [70], which is a bridge between Python and YAML , so we can use Python for programming automation, also there is automation editor on the own application and finally Node-RED application which is a flow-based development tool.
Home Assistant also works on the own hardware, not requiring any external connection (which adds stability to system). UI (Lovelce) is the most attractive of three, having an useful editor. Though Home Assistant have mobile applications development it is not so finished as OpenHAB, having the iOS application more or less correct, but really basic application on Android. Nonetheless, the philosophy of Home Assistant is not depending on applications, but just on the embedded web server.

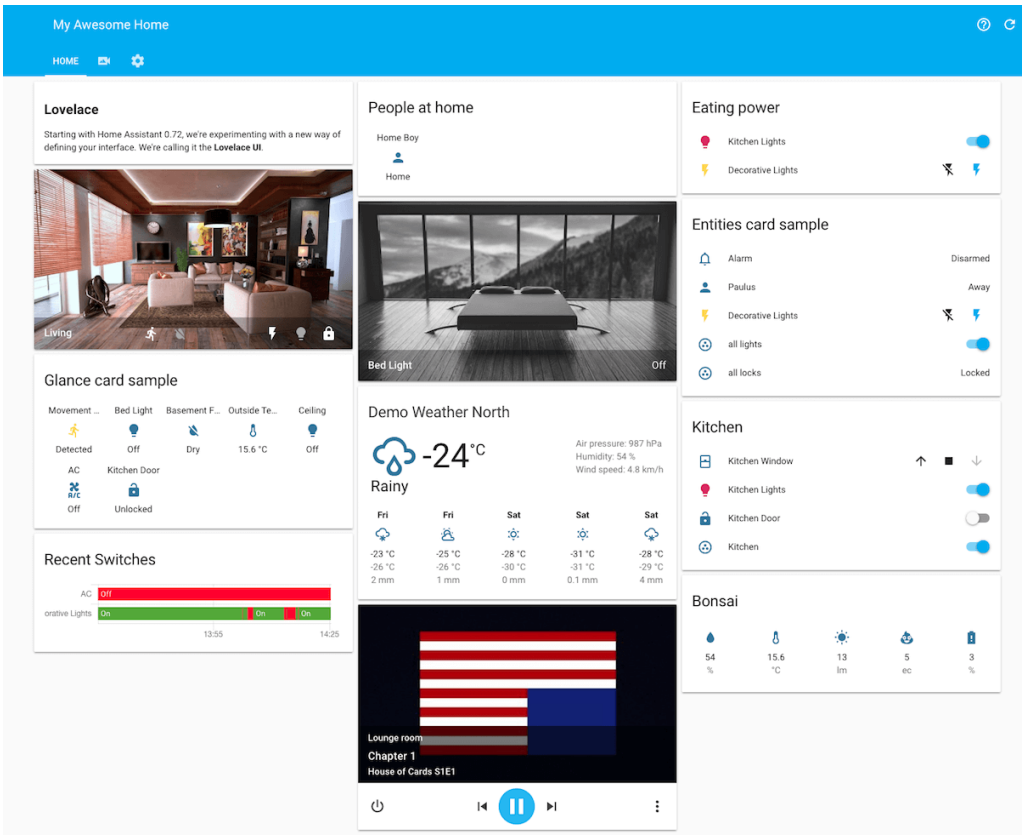


Figure 33 - Home Assistant User Interface [71].

3.3.2.3. Domoticz

Domoticz is written in C++ and founded by Gizmocuz (from Netherlands, not found the real name), and with multitude of co-developers [72]. C++ (GPLv3) programming make it efficient and avoid dependencies. It can be used on Raspberry Pi, Windows, Linux, Mac OS X and embedded devices. Its main feature is lightweight. User interface is a front end HTML5 automatically adapted to mobile devices. Another remarkable feature is accepting/sending push notifications from iPhone/Android (allowing to receive message from Telegram, between others). Designed for simplicity it allows sharing/using external devices. It has a smaller community than previous options, so documentation and forums are quite limited.

Installation is a little bit more complicated than previous options. To install Domoticz you first need to have a Raspberry Pi distribution like Raspbian installed and with Internet connectivity.

Configuration is done through web interface almost complete; no doubt, the most intuitive from all options.

Flexibility, Domoticz is very stable and will do the basics just fine, but quite limited in terms of supported devices and configurations.

Pace of development is the slowest of the different alternatives (it comes with the community size). Especially slow when it comes to new devices using not very known protocols; but having MQTT protocol any device can be integrated [73].

Automation is probably one of the strongest points of Domoticz [74], specifically how easy is to program an automation script using LUA (like Python) [75] and/or Blockly (graphical language based on functional blocks, for creating automations via web-GUI) [76].

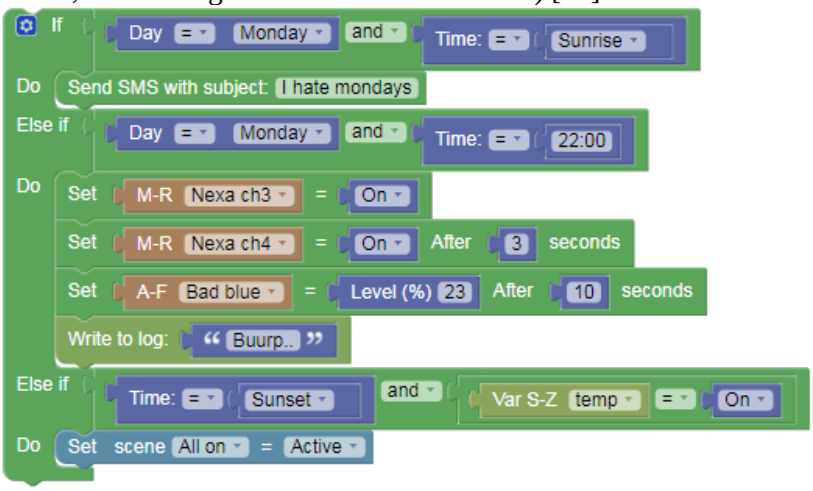


Figure 34 - Blockly example [74].

Domoticz is a great idea, though it seems that number of users is decaying with time. Great and easy to use for general purpose domotics, but lacking flexibility for specific SHEMS purposes. Good option for working with Z-Wave [77].

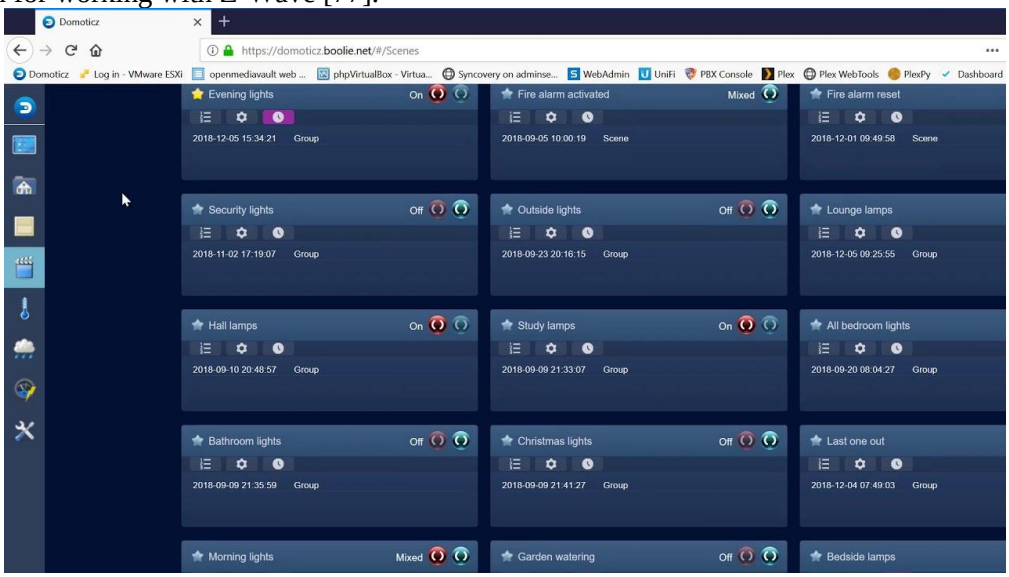


Figure 35 - Domoticz user interface [78].

3.3.2.4. ThingSpeak

ThingSpeak™ is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB® code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics [79]. Though it has been developed by MathWorks, it is open source code.

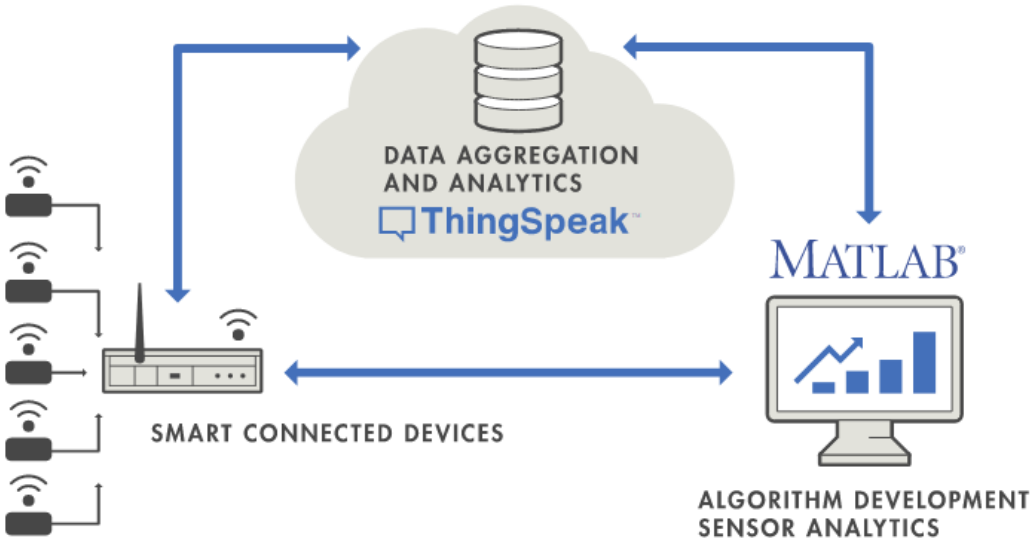


Figure 36 - Thingspeak working flow [79].

Above figure represents how Thingspeak works in the cloud, linking data aggregation with Matlab and Simulink tools for data analysis. Remark that cloud can be an external cloud or just a local area network.
Thingspeak uses MQTT and HTTP protocols for retrieving the data. One important thing to remark is that there is no need of of Matlab license from Matworks for Thingspeak users to analyze and visualize data.
Thingspeak works with: MATLAB, Arduino, Particle Photon and Electron, ESP8266 Wifi Module, Raspberry Pi, LoRaWAN, Things Network, Senet Libelium, Beckhoff, …
This application can share data with public channels.
Commercial users may sign up once for a time-limited free evaluation. All other commercial uses require a standard commercial license. Non-commercial users can use ThingSpeak for free subject to the limitations of the free license option. Users of the free option will be limited to sending no more than 3 million messages each year to the ThingSpeak service. Users of the free license will also be limited to 4 channels. For users of the free option, the message update interval limit remains limited at 15 seconds. Other limitations are described on the How to Buy pages [80].
So, though this application is interesting for research purposes, it is out of work focus, since, it is not low cost, and it is more intended for data analysis than controls, even when it can develop control functions. Nonetheless, it has been considered interesting to include, at least, a little remark of it, as an option for future developments.

3.3.2.5. OpenVPN

OpenVPN: It is an open source commercial application which adds the SHEMS chosen application a higher cybersecurity level. OpenVPN implements a virtual private network (VPN) to create secure point-to-point or site-to-site connections in routed or bridged configurations and remote access facilities [81].
OpenVPN can support 2 factor authentication (2FA) when combined with Google Authenticator; this is currently the best way for securing remote access to SHEMS Center.

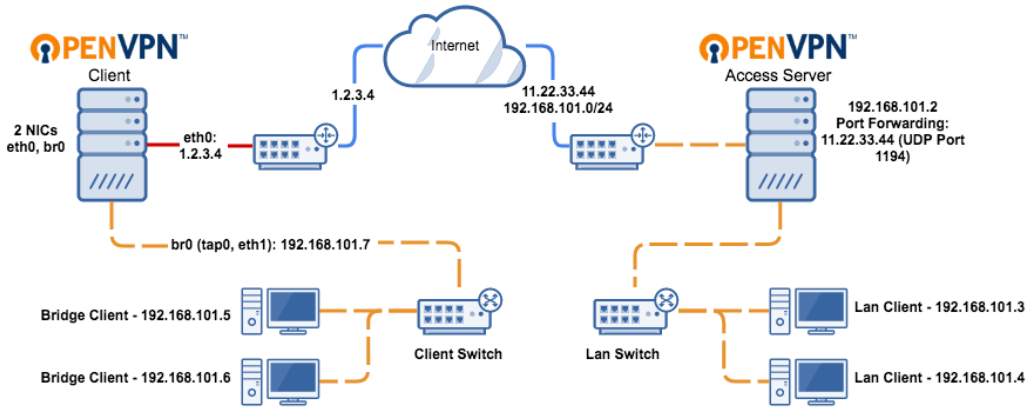


Figure 37 – OpenVPN [82].

4. METHODOLOGY - DOMOTICS COST EFFECTIVE ELEMENTS

In this chapter we will focus on basic functions on SHEMS to be develop by previously exposed tools.

4.1 Low Cost Domotics Application to SHEMS

It is going to be carried on a comparation between existing SHEMS systems and low cost domotics applications. There are many references in this field [83], [84] and [85] but In order to accomplish it we are going to focus on one of the main actual commercial company for SHEMS implementation: CarbonTRACK®.
CarbonTRACK® is an Australian company but founded in South Africa, committed to provide "Technology to monitor, control, automate and share energy".
As part of the offer of this company there are products and consulting. Of course, consulting is the real "added value" of the company, but as an exercise, let's see the equivalent in low cost domotics for the offered devices.
As part of the company's introduction, the following diagram is presented, to show the applications involved with CarbonTRACK:

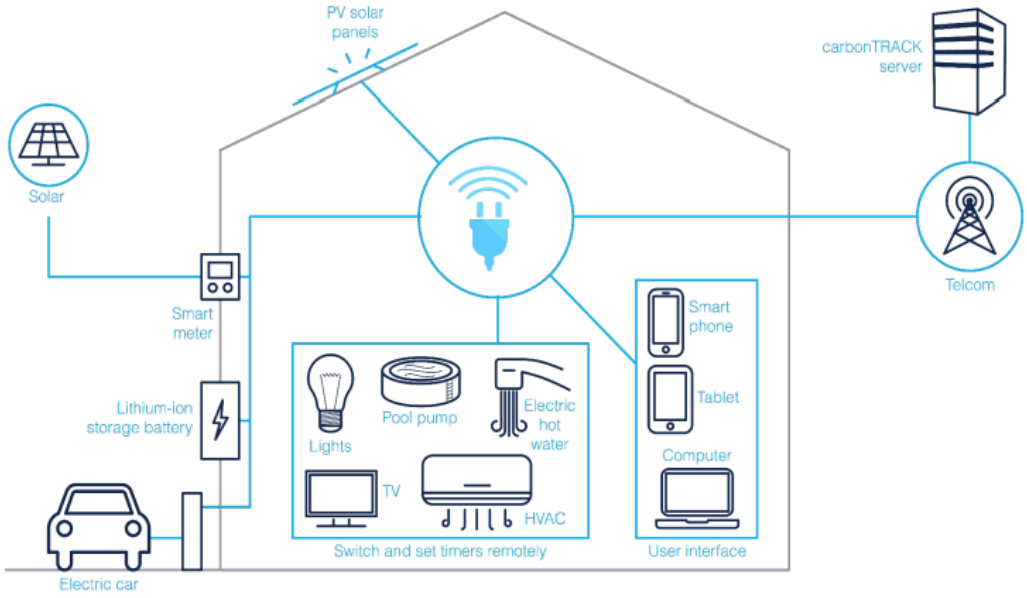


Figure 38 - CarbonTRACK applied in SHEMS [86].

As any classical SHEMS, is proposed [87]:

- Monitor device and appliance data.
- Minimum time frame data.
- Insights delivered: analysis of data to alarm user of abnormal consumpts.
- Turn devices on/off remotely.

- Set devices to operate on schedules.
- Manage the flow of energy from solar panels (or other renewable generators) to home or in and out the batteries.
- Allow machine learning to take over and run the system semi-automatically.

To develop this functions the following products, ZigBee based, are proposed:

- Smart-plug [88]: CarbonTRACK propose to use them with functions such as reduce vampire power, run appliances remotely, presence simulation and track power consume of devices/appliances. Datasheet can be found on appendix. They can be replaced by 4 € SonOff Basic devices in some cases or by 12 € SonOff POW R2 if it is wanted to monitor power. On previous scheme, smart plug will be used for turning on/off lights, pool pump, TV, and the same time that measuring the power for detecting vampire appliances and turn them off or stablish appliances patterns for monitoring state or discover high power requirement devices
- Additionally it can be added a contactor, CJX2-1810 7.69 €, if power consumption of the application is greater than 10 A (maximum allowed by SonOff Basic) and for controlling three phase loads, which coil is commanded by SonOff Basic. Also can be added a current transformer BZCT18AL-30a/5a, 2.60 € and small modify on SonOff POW R2 board to allow it for measuring currents higher than specified.

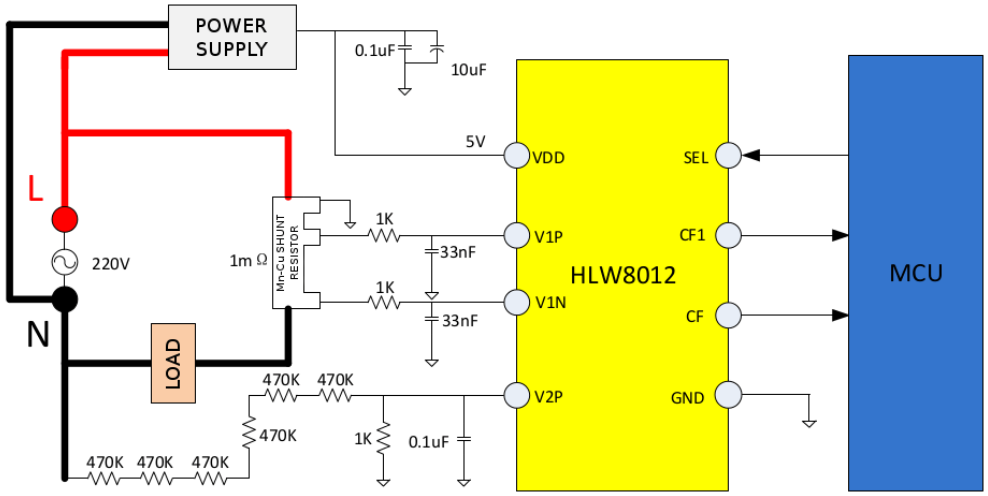


Figure 39 - HLW8012 element on SonOff Pow R2 [89].

Looking at above scheme, it will be just required to disconnect 1 mOhm resistor from the circuit and connect it to the current transformer secondary which primary will be measuring the current we need to monitor. Finally, a jumper will be done for continuing the circuit to supply load. Or a I2C current sensor can be added to SonOff TH [89].

Making above modifications will allow to control and monitor Solar PV, Battery and EV charging.

Climate command: For controlling HVAC (Heating, Ventilating and Air Conditioning) system it is intended this device which uses IR HVAC for control them. Datasheet can be found on appendix. It is proposed for getting the perfect temperature when arriving home (starting HVAC before arriving) and reducing climate bill. This function can be achieved using an IR Bridge (e.g. Eachen IR DC6 IR Bridge, 14.10 €) for controlling HVAC through infrared controller in order to adjust temperature, on/off and other functions which

firmware can be replaced by Tasmota and others. Additionally, SonOff TH10, 11.99 €, with temperature sensor will be added to monitor the temperature.

- Smart thermostat: Practically the same specifications than previous device, but adding a HMI in order to visualize and manage it locally. This same function can be achieved using smart phone with an APP installed, connected to SHEMS control center; but if it is required to have a local HMI for controlling; it can be based a display (1.83 €, HD44780 LCD w/CII/I2C adaptor) on I2C direct communication with SonOff TH10 and use of own buttons integrated with TH10. Or SunFounder Display Capacitive Touch Screen 7 Inch HDMI, 50 €, which will allow local control for SHEMS control center.
- SHEMS center: The solution of CarbonTRACK for SHEMS center is CT200i Smart Gateway. It is based on ZigBee and ZWave communications. The main features are: Big picture and granular data for energy monitoring, smart control, real time data collection, solar and battery maximization safety and security system alerts. The proposed low-cost option for this function is a Raspberry PI (model 3B will be enough) with Homeassitant, Domoticz or OpenHub software.

Finally, it is recommended to look on CarbonTRACK web, sections of case studies [90] and stories [91].

4.2 Basic example

This section will show just an overview of what can be done with the low-cost elements we have described.

Starting point will be a real case: one system of saline coloration has been installed on a pool. The system works converting NaCl in Cl by means of electrolysis. Such way, pool water is salt, so skin reactions are avoided, but at the same time, Cl level and pH is remained at levels for avoiding water corruption. The only problem is that the system needs water pump to be running for 3 hours every day. That is a high cost in the electricity bill. Solution: adopt hourly pricing tariff and use the pump at the cheapest hours considering that application can be classified as a schedulable and interruptible appliance.

For doing that, the information provided by official APP from eSiOS system (REE, Spainish national grid utility) will be used: [92]. This page provides information about hourly rate:

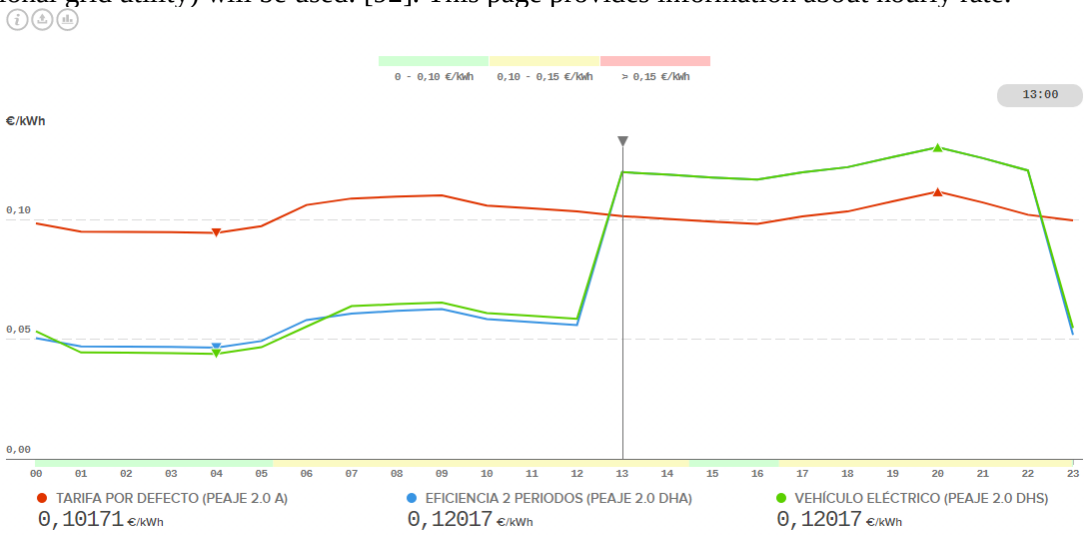


Figure 40 - Electricity hourly rate provided by official page [92].

Other utilities has similar applications, like Exelon's company ComEd [109]. This rate doesn't include power terms or taxes, but they are constant for every period, so, that choosing the cheapest period, we will have the cheapest energy. Proposed SHEMS system will call this page, get the cheapest 3 hours and turn on the pool pump on such hours. In above case, from 01.00 to 04.00.
The way in which the information is gathered from application is like connecting to a web page:
*https://apidatos.ree.es/es/datos/mercados/precios-mercados-tiempo-real?start_date=**2020-09-29T00:00**&end_date=**2020-09-29T23:59**&time_trunc=**hour***
Where bold-italic text are parameters set to get the desired answer. The output returned will be the hour price for the specified ranges and parameters.
To develop this function, HomeAssistant has been chosen and it will be installed on a RaspBerry PI 3B.
Firstly, install Raspberry PI OS Lite, formerly called Raspbian, following instructions on [93].
Next, install HomeAssistant, using instructions on [94]. Remark that is needed SHH connection to Raspberry, for Windows a program like Putty can be used; or directly with a Linux computer using SHH command.
Pool pump current is 4,5 A (400 VAC), but is three phase, so it is needed an interposing three phases contactor, coil commanded at 240 VAC single phase. Coil will be connected to a SonOff basic relay. So, next step will be to flash SonOff device with a customized firmware. It has been chosen Tasmota. Following the instructions on the link [95]. One USB/UART adapter can be required, but also it can be done, quite easily just using the own RaspBerry PI. Tasmota tutorial finish showing how to enable MQTT protocol.
Following step will be to add Mosquitto broker to RaspBerry PI. This will allow HomeAssistant to communicate with SonOff MQTT; for doing such, just is required to follow the next tutorial [96].
For continuing, it must be connected HomeAssistant with SonOff relay, following the instructions on this link [97] or [98]. At the end of this part there will be a new device added to Home Assistant, as showed in bellow picture:

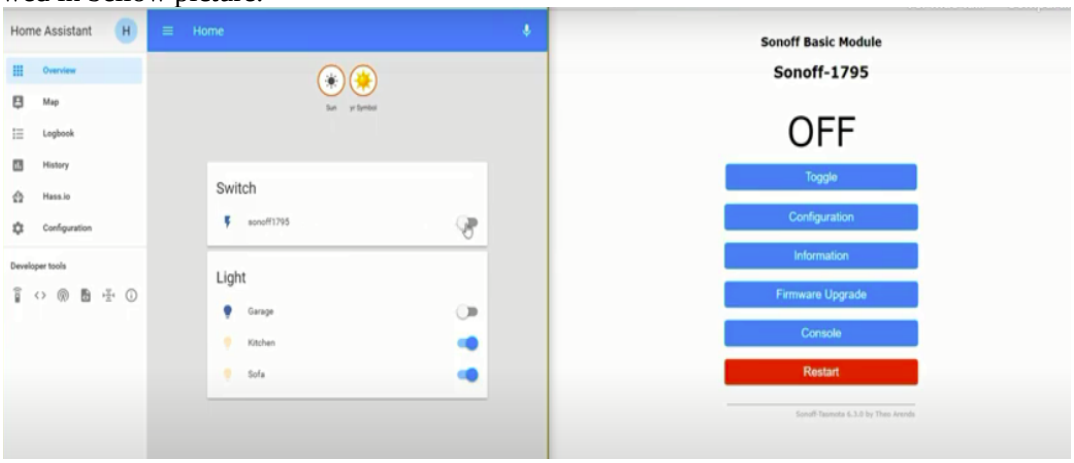


Figure 41 - Home Assistant SonOff as switch and Tasmota UI [99].

Next step is to add a "sensor" to Home Assistant, which is the interface in charge to obtain the lowest price of electricity. This sensor is called pvpc, is natively installed on Home Assistant (many versions ago), so it is just needed to call it and configure it. It is easy, but there are some instructions [100]. We just must set the tariff of reference, between three options: normal (PVPC), discrimination (2 periods) and electric car. Manual will be selected.

Sensor.pvpc is the name for electricity price. Entering in automation, it can be checked that program is intuitive.
Finally, it will be used DuckDNS service to allow remote access to our SHEMS centre. DuckDNS is a free service that assign a web domain to the dynamic IP of used modem. So, on firewall option of home modem, Raspberry MAC is assigned to a fix IP; on ports forwarding section (firewall), such fix IP is assigned to an external port. Easily, DuckDNS client is installed in Raspberry PI [101]. So, periodically, DuckDNS client is sending the dynamic IP to DuckDNS server, and when externally the assigned web domain is inserted in a web navigator, browser is address to the dynamic IP that in this moment the data provider company has assigned to our modem. After DNS domain we must set ":port" where port is the port assigned to RaspBerry in our modem. Now, there is access from remote to our Home Assistant server.
Economic analysis of the proposed example: 0.04660 €/kWh in valley rate to 0,13058 €/kWh in peak rate. Pump VICTORIA PLUS 3, with flow rate of 34000 litres/hour and rated power 2,4 kW, working 3 hours per day, potential savings are: (0,13058-0,04660)*2,4*3=0,60 € per day; 18,13 € per month.

SHEMS budget:	
Raspberry PI3B+	35,00 €
SonOff Basic	4,00 €
3 ph. Contactor	7,69 €
TOTAL:	**46,69 €**

Table 3: "SHEMS best price" project budged.

Potential savings:	
Worst hour rate (€/kWh)	0,13058
Better hour rate (€/kWh)	0,04660
Pump power (kW)	2,4
Hours per day	3,0
Savings/day	0,60 €
Days to cover budget	**77,21746**

Table 4: "SHEMS best price" Return Of Investment (ROI) analysis.

So, just using it for one basic SHEMS function, this system is economically interesting for mass users in their home, since investment return is 78 days. SHEMS Center (Raspberry PI 3B with installed applications) is available to develop further SHEMS functions, like PEV charging management, PV for home power supply, appliances control, home heating / air conditioning optimization, … and going one step beyond, complex smart algorithms like fuzzy logic or genetic neural networks can be implemented on RaspBerry PI 3B in order to forecast weather or user habits to optimize energy management.

5. CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions.

This paper shows how a complex SHEMS can be develop based on very low cost domotics; allowing SHEMS can be installed on every home. This is one of the key stones to reach Distributed Energy Renewable Energy Systems to become a reality; having all the small home generation systems interconnected allowing to manage them, not as isles, but as a big generation plant; allow to coordinate them as a global factor with existing generation plants and distribution and transport networks. But this goal, of a coordinated network of small "prosumagers" requires of a low-cost technology, since it must become attractive to common people. Even for small houses in which installing renewable energy is not possible, having installed a SHEMS system will allow to coordinate energy requirements and increase energy efficiency, allowing renewable generation plants to coordinate and obtain the maximum profit of natural energy resources, as wind, sun, geothermal, …

Summarizing: Renewable energy requires mandatorily of SHEMS systems, at a price which allows installing them massively.

5.2 Future works

Taking above conclusions as starting point, the following projects can be developed:

- Complete SHEMS including:
  - Battery charging management.
  - Electric Vehicle charging management.
  - Renewable energy management.
  - Power supply analysis, adding harmonics analysis
  - Loads analysis, with individual measure for each significant load, allowing to detect energy vampires or deviation on power finger print for appliances.
  - Home appliances control, further than on/off but also regulation, based on smart appliances but finding out how no-prepared appliances can be controlled, e.g. replacing push buttons by external contacts commanded by smart devices.
  - HVAC control.
  - Multi-points temperature measurement; communication network based on ZigBee or Z-Wave to allow battery supply [102], [103], [104], [105]
  - Hot water radiators smart controlled by electric valves for central heating control.
  - Motorized sunshades controlled by smart devices.
  - Lights controlled by smart devices.
- Based on above proposed SHEMS smart algorithms can be controlled using advanced algorithms. E.g.:
  - On [106] Different optimization algorithms are proposed in order to optimize PV renewable installation: Genetic algorithm, particle swam optimization and Artificial Bee Colony (ABC).
  - On [107] is presented a system for Energy Optimization providing adaptive heating, adaptive electricity, forecasting user status and real time

classification, based on machine learning (supervised learning), using Azure Machine Learning Platform.

- On [108] it is showed how, using Binary particle swarm optimisation with quadratic transfer function (quadratic binary particle optimization) algorithm appliances scheduling can be optimized.
- On [24] is presented an adaptive smart home system for optimal utilization of power based on genetic-fuzzy-neural networks technique for capturing a human behaviour patterns and use it to predict the user's mood.
- On [20] uses Non-Dominated Sorted Genetic Algorithm (NSGA-II), in order to determine the scheduling of home appliances for the time horizon.
- On [21] is proposed a greedy iterative algorithm (GIA) to consider the optimum start and running time for the devices so for each user, the expense feature is predicted and have been designed for restrictions.

- Finally, data aggregation (using Thingspeak, for example), compiling all data coming from all SHEMS allowing to electric utility better management of power generation and reducing outages. Even electric utility can offer to consumer the best energy prices for consumer give control of a certain amount of load to the electric utility: e.g. based on previous case, consumer gives control of the load of pool pump (2,4 kW * 3 hours/day=7,2 kWh/day), so utility can have a load 7,2 kWh for moving along the day in order to use the renewable energy when it is producing…

BIBLIOGRAPHY

[1] REN21 (Renewable Energy Policy Network for the 21st Century), "RENEWABLES 2016 GLOBAL STATUS REPORT," marzo 2015. [Online]. Available: https://www.ren21.net/gsr-2016/chapter03.php#i0. [Accessed julio 2020].

[2] M. d. C. Falante, "La energía del cambio, Soluciones tecnológicas para cambiar…," ABENGOA, febrero 2018. [Online]. Available: http://www.laenergiadelcambio.com/que-es-microgrid/. [Accessed julio 2020].

[3] Vortex Bladeless, "Reinventing Wind Turbines," september 2020. [Online]. Available: https://vortexbladeless.com/. [Accessed 26th september 2020].

[4] L. Jeong, C. Sic, P. Ki, H. Soo and L. Woo, "A study on the use cases of the smart grid home energy management system," *Proceedings of the international conference on ICT convergence (ICTC),* vol. p. 746–50, 2011.

[5] J. Han, C. Choi, W. Park and I. Lee, "Green home energy management system through comparison of energy usage between the same kinds of home appliances.," *Proceedings of the 15h IEEE international symposium on consumer electronics (ISCE),* Vols. p. 1-4, 2011.

[6] e. a. B. Zhou, "Renewable and Sustainable Energy Reviews 61," Vols. p. 30-40, 2016.

[7] Rad MAH, Wong VWS, Jatskevich J, Schober R and Garcia AL, "Autonomous demand side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Trans Smart Grid,* vol. 1(3):302–31.

[8] A. Dimeas, S. Drenkard, N. Hatziargyriou, S. Karnouskos, K. Kok, J. Ringelstein and A. Weidlich, "Smart houses in the smart grid: developing an interactive network," *IEEE Electr Mag,* Vols. p. 81-93, 2014.

[9] J. Zeng, D. Gao and L. Lin, "Smart meters in smart grid: an overview," *Proceedings of the green technologies conference,* Vols. p. 57-64, 2013.

[10] A. Diker and E. AVCI, "Feature Extraction of ECG Signal by using Deep Feature," in *2019 7th International Symposium on Digital Forensics and Security (ISDFS),*, Barcelos, Portugal. DOI: 10.1109/ISDFS.2019.8757522, 2019.

[11] Z. Zhao, W. Lee, Y. Shin and K. Song , "An optimal power scheduling method for demand response in home energy management system," *IEEE Trans Smart Grid,* Vols. 4(3):1391-400, 2013.

[12] Y. Zong, D. Kullmann, A. Thalov, O. Gehrke and H. Bindner, "Application of model predictive control for active load management in a distributed power system with high wind penetration," *IEEE Trans Smart Grid,* vol. 3(2):1055–62, 2012.

[13] Nguyen DT and Le LB, "Joint optimization of electric vehicle and home energy scheduling considering user comfort preference," *IEEE Trans Smart Grid,* vol. 5(1):188–99, 2014.

[14] Pedrasa MAA, Spooner TD and MacGill IF, "Coordinated scheduling of residential distributed energy resources to optimize smart home energy services," *IEEE Trans Smart Grid,* vol. 1(2):134–43, 2010.

[15] Rad AHM and Garcia AL, "Optimal residential load control with price prediction in real-time electricity pricing environments," *IEEE Trans Smart Grid,* vol. 1 (2):120–33, 2010.

[16] Corno F and Razzak F, "Intelligent energy optimization for user intelligible goals in smart home environments," *IEEE Trans Smart Grid,* vol. 3(4):2128–35, 2012.

[17] Chen XD, Wei TQ and Hu SY, "Uncertainty-aware household appliance scheduling considering dynamic electricity pricing in smart home," *IEEE Trans Smart Grid,* vol. 4(2):932–41, 2013.

[18] Konstantinos O, Emmanouil A and Charis S, "Frequency-based control of islanded microgrid with renewable energy sources and energy storage," *J. Mod. Power Syst Clean Energy,* vol. 4(1):54–62, 2016.

[19] Chen C, Wang JH and Kishore S, "A distributed direct load control approach for large-scale residential demand response," *IEEE Trans Power Syst,* vol. 29 (5):2219–28, 2014.

[20] Veras, J.M., Silva, I.R.S, Pinheiro, P.R., Rabêlo, R.A.L., Veloso, A.F.S., Borges, F.A.S. and Rodrigues, J.J.P.C., "A Multi-Objective Demand Response Optimization Model for Scheduling Loads in a Home Energy Management System," *Sensors,* Vols. 18, 3207, no. DOI: 10.3390/s18103207, 2018.

[21] Ma, Y and Li, B., "Hybridized Intelligent Home Renewable Energy Management System for Smart Grids," *Sustainability,* Vols. 12, 2117, no. DOI: 10.3390/su12052117, 2020.

[22] Ozturk Y, Senthilkumar D, Kumar S and Lee G, "An intelligent home energy management system to improve demand response," *IEEE Trans Smart Grid,* vol. 4(2):694–701, 2013.

[23] Hassan abdolrezaei, Mohammad hassan moradi and Mohammad javad rastegar fatemi, "Short-term Load Forecasting Using Genetic- Fuzzy Algorithm and Neural Network," no. COI Paper: ELEMECHCONF03_0197.

[24] Anwar Jarndal, "SMART HOME SYSTEM FOR ENERGY SAVING USING GENETIC-FUZZY-NEURAL NETWORKS APPROACH," *INTERNATIONAL JOURNAL OF SUSTAINABLE WATER AND ENVIRONMENTAL SYSTEMS (SWES),* Vols. 08, ISSUE 1, p.27-31, no. DOI: 10.5383/swes.8.01.006.

[25] Microcontrollers Lab, "ESP8266 pinout reference and how to use GPIO pins," [Online]. Available: https://microcontrollerslab.com/esp8266-pinout-reference-gpio-pins/. [Accessed 1st september 2020].

[26] DIY Projects, "ESP01. Get started with the Arduino or PlatformIO IDE. Which module to choose? Pinout," 18th september 2020. [Online]. Available: https://diyprojects.io/esp01-get-started-arduino-platformio-ide-module-choose-pinout/. [Accessed 26th september 2020].

[27] Online Shouter, "TOP 5 WAYS TO WIRE UP ESP8266 MODULE TO YOUR ARDUINO UNO BOARD," 22nd september 2017. [Online]. Available: http://onlineshouter.com/top-5-ways-wire-esp8266-module-arduino-uno-board/. [Accessed 26th september 2020].

[28] Alphabet Inc., "Google LLC - Generic search for images," [Online]. Available: https://www.google.com/.

[29] ACopTex, "Basics: Project 069b Sonoff smart wifi switch - flashing a custom firmware," 2019. [Online]. Available: http://acoptex.com/project/311/basics-project-069b-sonoff-smart-wifi-switch-flashing-a-custom-firmware-at-acoptexcom/. [Accessed 2020].

[30] ESPRESSIF, "ESP32 Overview," 24th September 2020. [Online]. Available: https://www.espressif.com/en/products/socs/esp32/overview. [Accessed 24th September 2020].

[31] Espressif Systems (Shanghai) CO., LTD, "API Reference » Application Protocols » HTTPS server," 2020. [Online]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_https_server.html. [Accessed september 2020].

[32] V. I. Lucian, "GeeksTips.com, ESP32 – CHEAPEST IOT WIFI AND BLUETOOTH

READY MODULE," [Online]. Available: https://www.geekstips.com/esp32-review-idf-programming-tutorial/. [Accessed 2020].

[33] D. No, “ESPloradores,” 30 April 2019. [Online]. Available: https://www.esploradores.com/eligiendo_una_placa_esp32/. [Accessed 24th September 2020].

[34] Wikipedia, the free encyclopedia, “ESP32,” 23rd march 2020. [Online]. Available: https://en.wikipedia.org/wiki/ESP32. [Accessed june 2020].

[35] Jithendra, “Ramdon Nerd Tutorials,” 29th December 2016. [Online]. Available: https://randomnerdtutorials.com/getting-started-with-esp32/. [Accessed 24th September 2020].

[36] Espressif Systems, “The Internet of Things with ESP32,” 24th September 2020. [Online]. Available: http://esp32.net/. [Accessed 24th September 2020].

[37] Wikipedia, the free encyclopedia, “LoRa,” 14th September 2020. [Online]. Available: https://en.wikipedia.org/wiki/LoRa. [Accessed 23rd September 2020].

[38] S. Santos, “Maker Advisor,” 18 May 2020. [Online]. Available: https://makeradvisor.com/esp32-development-boards-review-comparison/. [Accessed 24th Septemeber 2020].

[39] J. Beningo, “DigiKey - How to Select and Use the Right ESP32 Wi-Fi/Bluetooth Module for an Industrial IoT Application,” 21st January 2020. [Online]. Available: https://www.digikey.es/en/articles/how-to-select-and-use-the-right-esp32-wi-fi-bluetooth-module. [Accessed 14th July 2020].

[40] Espressif System, "Espressif Modules," 23rd August 2020. [Online]. Available: https://www.espressif.com/en/products/modules. [Accessed 24th September 2020].

[41] Espressif Systems, “Espressif Resources,” 24th September 2020. [Online]. Available: https://www.espressif.com/en/products/socs/esp32/resources. [Accessed 24th September 2020].

[42] TME Electronic Components, "Transceptores inalámbricos de SISTEMAS ESPRESSIF," 24th September 2020. [Online]. Available: https://www.tme.eu/html/ES/modulos-y-transceptores-wireless-de-la-marca-espressif/ramka_17873_ES_pelny.html. [Accessed 21st April 2020].

[43] Ó. Gutiérrez, M. Prieto, A. Gómez and M. Sánchez-Reyes, "TID characterization of COTS parts using Radiotherapy Linear Accelerators," *IEICE Electronics Express*, vol. 16. DOI: 10.1587/elex.16.20190077. , 2019.

[44] The Pi4J Project, "Pin Numbering - Raspberry Pi Zero W," 5th march 2019. [Online]. Available: https://www.pi4j.com/1.2/pins/model-zerow-rev1.html. [Accessed 2020].

[45] Raspberry Pi Foundation, “About us,” September 2020. [Online]. Available: https://www.raspberrypi.org/about/. [Accessed 23rd September 2020].

[46] Wikipedia, the free encyclopedia, "Raspberry Pi," 27th september 2020. [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi#cite_note-13. [Accessed 27th september 2020].

[47] WatElectronics.com, "Know all about Raspberry Pi Board Technology," 26th july 2019. [Online]. Available: https://www.watelectronics.com/know-all-about-raspberry-pi-board-technology/. [Accessed 12th july 2020].

[48] Let's Control It, "letscontrolit/ESPEasy," 2020, [Online]. Available: https://github.com/letscontrolit/ESPEasy.

[49] RUTG3R, "SONOFF FIRMWARE TUTORIAL TO ESP EASY," 2nd february 2017. [Online]. Available: https://rutg3r.com/sonoff-firmware-tutorial-to-esp-easy/. [Accessed 2020].

[50] Lets Control It, "EasyProtocols," 18th january 2018. [Online]. Available: https://www.letscontrolit.com/wiki/index.php?title=EasyProtocols. [Accessed 27th september 2020].

[51] Tasmota, "Getting Started," [Online]. Available: https://tasmota.github.io/docs/Getting-Started/. [Accessed 27th september 2020].

[52] Blakadder, "Tasmota Device Templates Repository," [Online]. Available: https://templates.blakadder.com/index.html. [Accessed 27th september 2020].

[53] Domotica en casa, "Tutorial: Flashea Tasmota en un dispositivo Sonoff (o un ESP8266) con Arduino IDE," [Online]. Available: https://domoticaencasa.es/tasmota-sonoff-esp8266-arduino-ide/. [Accessed 27th september 2020].

[54] X. Pérez, "ESPurna Firmware," 27th september 2020. [Online]. Available: https://github.com/xoseperez/espurna/wiki. [Accessed 27th september 2020].

[55] X. Pérez, "Supported Hardware," 26th september 2020. [Online]. Available: https://github.com/xoseperez/espurna/wiki/Hardware. [Accessed 27th september 2020].

[56] L. d. V. Hernández, "Domótica con relé Sonoff WiFi ESP8266 y ESPurna," [Online]. Available: https://programarfacil.com/esp8266/domotica-sonoff-wifi-espurna/. [Accessed 27th september 2020].

[57] AWS, "AWS, Free RTOS," 22nd March 2020. [Online]. Available: https://docs.aws.amazon.com/es_es/freertos/latest/userguide/burn-initial-firmware-esp.html. [Accessed 24th September 2020].

[58] Espruino, "Espruino on ESP32," 22nd May 2020. [Online]. Available: https://www.espruino.com/ESP32. [Accessed 24th September 2020].

[59] MicroPython, "Firmware for Generic ESP32 module," 2018. [Online]. Available: https://micropython.org/download/esp32/. [Accessed 24th September 2020].

[60] Tasmota, "ESP32 Support (Beta development)," 15th June 2020. [Online]. Available: https://tasmota.github.io/docs/ESP32/. [Accessed 21st August 2020].

[61] Lets Control It, "ESPEasy32," 30th March 2018. [Online]. Available: https://www.letscontrolit.com/wiki/index.php/ESPEasy32. [Accessed 24th September 2020].

[62] ESPHome, "Migrating from Sonoff Tasmota," [Online]. Available: https://esphome.io/guides/migrate_sonoff_tasmota.html. [Accessed 17th February 2020].

[63] Domotica en casa, "ESPHome: Instalación y primera prueba con ESP32 (o ESP8266)," 23rd August 2019. [Online]. Available: https://domoticaencasa.es/esphome-tutorial-instalacion-esp32-esp8266/. [Accessed 12th December 2019].

[64] HomeOps, "Tasmota vs ESPurna vs ESPEasy - overview," 8th January 2018. [Online]. Available: https://lobradov.github.io/FOSS-Firmware-comparison-overview/. [Accessed 27th september 2020].

[65] U. N. G, "ESPurna, Tasmota o ESPEasy para dispositivos ESP8266," 7th April 2019. [Online]. Available: https://www.alferez.es/iot/espurna-tasmota-o-espeasy-para-dispositivos-esp8266/. [Accessed 17th September 2020].

[66] Aprendiendo Arduino, "MQTT," [Online]. Available: https://aprendiendoarduino.wordpress.com/tag/mqtt-qos/. [Accessed 28th September 2020].

[67] OpenHAB, "openHAB Cloud Connector," [Online]. Available: https://www.openhab.org/addons/integrations/openhabcloud/. [Accessed 28th September 2020].

[68] OpenHAB, “openHAB UIs,” [Online]. Available: https://www.openhab.org/docs/tutorial/uis.html. [Accessed 28th September 2020].

[69] P. Schoutsen, “Perfect Home Automation,” 19th January 2016. [Online]. Available: https://www.home-assistant.io/blog/2016/01/19/perfect-home-automation/. [Accessed 28th September 2020].

[70] APP Daemon, “AppDaemon Tutorial for HASS Users,” 2019. [Online]. Available: https://appdaemon.readthedocs.io/en/latest/HASS_TUTORIAL.html#how-it-works. [Accessed 28th September 2020].

[71] P. Schoutsen, "0.72: Lovelace UI, KIWI Doorlocks, Wireless Tags, Insteon X10.," 22nd June 2018. [Online]. Available: https://www.home-assistant.io/blog/2018/06/22/release-72/. [Accessed 28th September 2020].

[72] Domoticz, “GitHub,” 23rd September 2020. [Online]. Available: https://github.com/domoticz/domoticz/graphs/contributors. [Accessed 23rd September 2020].

[73] Domoticz, "MQTT," 10th February 2020. [Online]. Available: https://www.domoticz.com/wiki/MQTT. [Accessed 29th September 2020].

[74] Domoticz, "Automation," 12th August 2020. [Online]. Available: https://www.domoticz.com/wiki/Automation#Blocky_designs. [Accessed 29th September 2020].

[75] Domoticz, “LUA commands,” 29th August 2020. [Online]. Available: https://www.domoticz.com/wiki/LUA_commands. [Accessed 29th September 2020].

[76] Domoticz, “Blockly,” 19th August 2019. [Online]. Available: https://www.domoticz.com/wiki/Blockly. [Accessed 29th September 2020].

[77] Domoticz, "Zwave," 5th August 2018. [Online]. Available: https://www.domoticz.com/wiki/Zwave. [Accessed 29th September 2020].

[78] J. Thompson, “YouTube - Domoticz UI basics,” 5th Decemeber 2018. [Online]. Available: https://www.youtube.com/watch?v=T8PWz7ZwTMs&app=desktop. [Accessed 29th September 2020].

[79] The MathWorks, Inc, “ThingSpeak,” 23rd September 2020. [Online]. Available: https://thingspeak.com/pages/learn_more. [Accessed 23rd September 2020].

[80] The MathWorks, Inc., “ThingSpeak,” 23rd September 2020. [Online]. Available: https://thingspeak.com/pages/license_faq. [Accessed 23rd September 2020].

[81] Wikipedia the free enciclopedia, “OpenVPN,” 26th July 2020. [Online]. Available: https://en.wikipedia.org/wiki/OpenVPN. [Accessed 23rd September 2020].

[82] OpenVPN, "Site-To-Site Layer 2 Bridging Using OpenVPN Access Server And A Linux Gateway Client," [Online]. Available: https://openvpn.net/vpn-server-resources/site-to-site-layer-2-bridging-using-openvpn-access-server/. [Accessed 29th September 2020].

[83] BEMI Automation, “Key Benefits of KNX Automation for Commercial Buildings,” [Online]. Available: https://www.bemi.fi/key-benefits-of-knx-automation-for-commercial-and-industrial-buildings/ . [Accessed 29th September 2020].

[84] U.S. Department of Energy, “Buildings: Sensors and Controls,” [Online]. Available: https://www.energy.gov/eere/buildings/sensors-and-controls. [Accessed 29th September

2020].

[85] A. Potter, “Home energy management systems,” Choice, 11th September 2017. [Online]. Available: https://www.choice.com.au/home-improvement/energy-saving/reducing-your-carbon-footprint/articles/home-energy-management-systems. [Accessed 29th September 2020].

[86] Solar Choice, “carbonTRACK: Aussie-based startup delivers smarter solar, storage & energy use,” 30th November 2015. [Online]. Available: https://www.solarchoice.net.au/blog/carbontrack-smart-solar-storage-energy-use. [Accessed 29th September 2020].

[87] CarbonTrack, "What is a home energy management system?," 24th September 2020. [Online]. Available: https://carbontrack.com.au/blog/what-is-a-home-energy-management-system/. [Accessed 24th September 2020].

[88] CarbonTrack, "Smart plug," 24th September 2020. [Online]. Available: https://carbontrack.com.au/products/smart-plug/. [Accessed 24th September 2020].

[89] X. Pérez, "Tinkerman Power monitoring with Sonoff TH and ADC121," 25 January 2017. [Online]. Available: https://tinkerman.cat/post/power-monitoring-sonoff-th-adc121/. [Accessed 14 July 2020].

[90] CarbonTrack, "Case Studies," 24th September 2020. [Online]. Available: https://carbontrack.com.au/case-studies/. [Accessed 24th September 2020].

[91] CarbonTrack, "Stories," 24th September 2020. [Online]. Available: https://carbontrack.com.au/stories/. [Accessed 24th September 2020].

[92] Red Electrica de España, “PVPC esios,” 24th September 2020. [Online]. Available: https://www.endesaclientes.com/precio-luz-pvpc.html?d=Any. [Accessed 24th September 2020].

[93] Rapsberry Pi Foundation, “Raspberry Pi OS (previously called Raspbian),” 24th September 2020. [Online]. Available: https://www.raspberrypi.org/downloads/raspberry-pi-os/. [Accessed 24th September 2020].

[94] HomeAssistant, “Manual installation on a Raspberry Pi,” 24th September 2020. [Online]. Available: https://www.home-assistant.io/docs/installation/raspberry-pi/. [Accessed 24th September 2020].

[95] Arendst Tasmota, "Tasmota Getting Started," 24th September 2020. [Online]. Available: https://tasmota.github.io/docs/Getting-Started/. [Accessed 24th September 2020].

[96] Home Assistant, "MQTT," 24th September 2020. [Online]. Available: https://www.home-assistant.io/integrations/mqtt/. [Accessed 24th September 2020].

[97] Arendst Tasmota, "Tasmota Home Assistant," 24th September 2020. [Online]. Available: https://tasmota.github.io/docs/Home-Assistant/. [Accessed 24th September 2020].

[98] Matthew, "Automation Fixation," 6th January 2019. [Online]. Available: http://automation.moebius.site/2019/01/hassio-installing-a-sonoff-switch-tasmota-firmware/. [Accessed 24th September 2020].

[99] Automation Fixation, “You Tube - Hassio / Home Assistant: Installing a Sonoff Switch - Emulation and MQTT,” 14th January 2019. [Online]. Available: https://www.youtube.com/watch?v=4H8ALCaAb-I. [Accessed 29th September 2020].

[100] Home Assistant, “Spain electricity hourly pricing (PVPC),” 24th September 2020. [Online]. Available: https://www.home-assistant.io/integrations/pvpc_hourly_pricing/#tariff. [Accessed 24th September 2020].

[101] DuckDNS, "Install," 24th September 2020. [Online]. Available: https://www.duckdns.org/install.jsp. [Accessed 24th September 2020].

[102] Tasmota, "Zigbee," [Online]. Available: https://tasmota.github.io/docs/Zigbee/. [Accessed 29th September 2020].

[103] J.-L. A. (CNXSOFT), "Sonoff Zigbee Bridge Now Supports Tasmota Firmware, Home Assistant, Zigbee2Tasmota," 27th July 2020. [Online]. Available: https://www.cnx-software.com/2020/07/27/sonoff-zigbee-bridge-tasmota-firmware-home-assistant-zigbee2mqtt/. [Accessed 29th September 2020].

[104] Eduardo, "Sonoff ZBBridge, el nuevo gateway Zigbee que parece traer nuevos dispositivos," Domotica en casa, 21st April 2020. [Online]. Available: https://domoticaencasa.es/sonoff-zbbridge-gateway-zigbee/. [Accessed 29th September 2020].

[105] X. Pérez, "ESPurna firmware on Sonoff ZBBridge (Sonoff Zigbee Bridge)," 16th April 2020. [Online]. Available: https://github.com/xoseperez/espurna/issues/2224. [Accessed 29th September 2020].

[106] D. A. Dalgo Lascano, "Proyecto de una aplicación HEMS para la gestión de energía de hogares con generación de energía fotovoltaica situados en Ecuador," 2019.

[107] Logan Odell, Va Banh, Sean O'Hara and Waleng V, "Home energy management system end of project documentation," 2014.

[108] A. Jordehi, "Binary particle swarm optimisation with quadratic transfer function: A new binary optimisation algorithm for optimal scheduling of appliances in smart homes," *Applied Soft Computing Journal*, no. PII: S1568-4946(19)30112-7, 2019.

[109] ComEd, "5-MINUTE PRICE DATA API," [Online]. Available: https://hourlypricing.comed.com/hp-api/. [Accessed 1 October 2020].

A P P E N D I X

APPENDIX A : ESP8266EX DATA SHEET

ESP8266EX

Datasheet

Version 6.5
Espressif Systems

About This Guide

This document introduces the specifications of ESP8266EX.

Release Notes

Date	Version	Release Notes
2015.12	V4.6	Updated Chapter 3.
2016.02	V4.7	Updated Section 3.6 and Section 4.1.
2016.04	V4.8	Updated Chapter 1.
2016.08	V4.9	Updated Chapter 1.
2016.11	V5.0	Added Appendix II “Learning Resources”.
2016.11	V5.1	Changed the power consumption during Deep-sleep from 10 μA to 20 μA in Table 5-2.
2016.11	V5.2	Changed the crystal frequency range from “26 MHz to 52 MHz” to “24 MHz to 52 MHz” in Section 3.3.
2016.12	V5.3	Changed the minimum working voltage from 3.0 V to 2.5 V.
2017.04	V5.4	Changed chip input and output impedance from 50 Ω to 39 + j6 Ω.
2017.10	V5.5	Updated Chapter 3 regarding the range of clock amplitude to 0.8 V ~ 1.5 V.
2017.11	V5.6	Updated VDDPST from 1.8 V ~ 3.3 V to 1.8 V ~ 3.6 V.
2017.11	V5.7	• Corrected a typo in the description of SDIO_DATA_0 in Table 2-1; • Added the testing conditions for the data in Table 5-2.
2018.02	V5.8	• Updated Wi-Fi protocols in Section 1.1; • Updated description of the integrated Tensilica processor in 3.1.

Date	Version	Release Notes
2018.09	V5.9	• Update document cover; • Added a note for Table 1-1; • Updated Wi-Fi key features in Section 1.1; • Updated description of the Wi-Fi function in 3.5; • Updated pin layout diagram; • Fixed a typo in Table 2-1; • Removed Section AHB and AHB module; • Restructured Section Power Management; • Fixed a typo in Section UART; • Removed description of transmission angle in Section IR Remote Control; • Other optimization (wording).
2018.11	V6.0	• Added an SPI pin in Table 4-2; • Updated the diagram of packing information.
2019.08	V6.1	Removed description of the GPIO function in Section 4.1.
2019.08	V6.2	Updated notes on CHIP_EN in Section 5.1
2019.12	V6.3	Add feedback links.
2020.04	V6.4	• Removed the description of “Antenna diversity”; • Updated the feedback links.
2020.07	V6.5	• Updated the description of HSPI in Section 4.3; • Updated links in Appendix.

Documentation Change Notification

Espressif provides email notifications to keep customers updated on changes to technical documentation. Please subscribe at https://www.espressif.com/en/subscribe.

Certification

Download certificates for Espressif products from https://www.espressif.com/en/certificates.

Table of Contents

1. Overview

Espressif's ESP8266EX delivers highly integrated Wi-Fi SoC solution to meet users' continuous demands for efficient power usage, compact design and reliable performance in the Internet of Things industry.

With the complete and self-contained Wi-Fi networking capabilities, ESP8266EX can perform either as a standalone application or as the slave to a host MCU. When ESP8266EX hosts the application, it promptly boots up from the flash. The integrated high-speed cache helps to increase the system performance and optimize the system memory. Also, ESP8266EX can be applied to any microcontroller design as a Wi-Fi adaptor through SPI/SDIO or UART interfaces.

ESP8266EX integrates antenna switches, RF balun, power amplifier, low noise receive amplifier, filters and power management modules. The compact design minimizes the PCB size and requires minimal external circuitries.

Besides the Wi-Fi functionalities, ESP8266EX also integrates an enhanced version of Tensilica's L106 Diamond series 32-bit processor and on-chip SRAM. It can be interfaced with external sensors and other devices through the GPIOs. Software Development Kit (SDK) provides sample codes for various applications.

Espressif Systems' Smart Connectivity Platform (ESCP) enables sophisticated features including:

- Fast switch between sleep and wakeup mode for energy-efficient purpose;
- Adaptive radio biasing for low-power operation
- Advance signal processing
- Spur cancellation and RF co-existence mechanisms for common cellular, Bluetooth, DDR, LVDS, LCD interference mitigation

1.1. Wi-Fi Key Features

- 802.11 b/g/n support
- 802.11 n support (2.4 GHz), up to 72.2 Mbps
- Defragmentation
- 2 x virtual Wi-Fi interface
- Automatic beacon monitoring (hardware TSF)
- Support Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode

1.2. Specifications

Table 1-1. Specifications

Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4 GHz ~ 2.5 GHz (2400 MHz ~ 2483.5 MHz)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: –91 dbm (11 Mbps)
		802.11 g: –75 dbm (54 Mbps)
		802.11 n: –72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5 V ~ 3.6 V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	–40 °C ~ 125 °C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

📖 ***Note:***

The TX power can be configured based on the actual user scenarios.

1.3. Applications

- Home appliances
- Home automation
- Smart plugs and lights
- Industrial wireless control
- Baby monitors
- IP cameras
- Sensor networks
- Wearable electronics
- Wi-Fi location-aware devices
- Security ID tags
- Wi-Fi position system beacons

2. Pin Definitions

Figure 2-1 shows the pin layout for 32-pin QFN package.

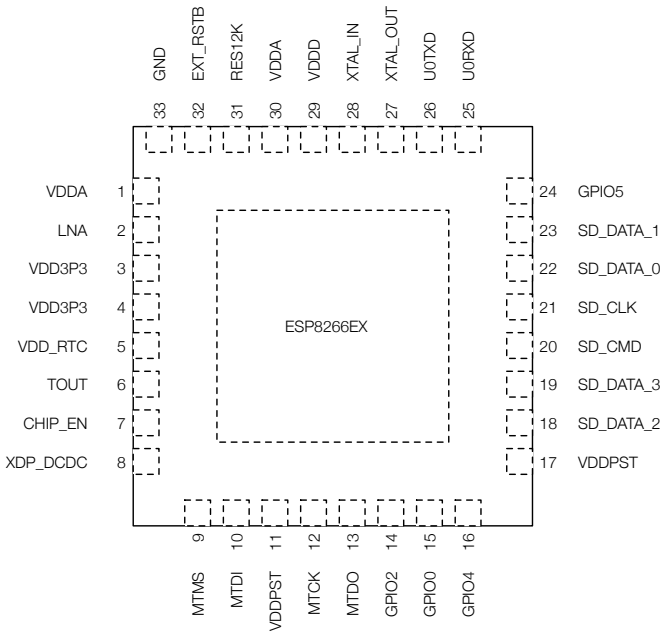


Figure 2-1. Pin Layout (Top View)

Table 2-1 lists the definitions and functions of each pin.

Table 2-1. ESP8266EX Pin Definitions

Pin	Name	Type	Function
1	VDDA	P	Analog Power 2.5 V ~ 3.6 V
2	LNA	I/O	RF antenna interface Chip output impedance = 39 + j6 Ω. It is suggested to retain the π-type matching network to match the antenna.
3	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V
4	VDD3P3	P	Amplifier Power 2.5 V ~ 3.6 V
5	VDD_RTC	P	NC (1.1 V)
6	TOUT	I	ADC pin. It can be used to test the power-supply voltage of VDD3P3 (Pin3 and Pin4) and the input power voltage of TOUT (Pin 6). However, these two functions cannot be used simultaneously.
7	CHIP_EN	I	Chip Enable High: On, chip works properly Low: Off, small current consumed

Pin	Name	Type	Function
8	XPD_DCDC	I/O	Deep-sleep wakeup (need to be connected to EXT_RSTB); GPIO16
9	MTMS	I/O	GPIO 14; HSPI_CLK
10	MTDI	I/O	GPIO 12; HSPI_MISO
11	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
12	MTCK	I/O	GPIO 13; HSPI_MOSI; UART0_CTS
13	MTDO	I/O	GPIO 15; HSPI_CS; UART0_RTS
14	GPIO2	I/O	UART TX during flash programming; GPIO2
15	GPIO0	I/O	GPIO0; SPI_CS2
16	GPIO4	I/O	GPIO4
17	VDDPST	P	Digital/IO Power Supply (1.8 V ~ 3.6 V)
18	SDIO_DATA_2	I/O	Connect to SD_D2 (Series R: 20 Ω); SPIHD; HSPIHD; GPIO9
19	SDIO_DATA_3	I/O	Connect to SD_D3 (Series R: 200 Ω); SPIWP; HSPIWP; GPIO10
20	SDIO_CMD	I/O	Connect to SD_CMD (Series R: 200 Ω); SPI_CS0; GPIO11
21	SDIO_CLK	I/O	Connect to SD_CLK (Series R: 200 Ω); SPI_CLK; GPIO6
22	SDIO_DATA_0	I/O	Connect to SD_D0 (Series R: 200 Ω); SPI_MISO; GPIO7
23	SDIO_DATA_1	I/O	Connect to SD_D1 (Series R: 200 Ω); SPI_MOSI; GPIO8
24	GPIO5	I/O	GPIO5
25	U0RXD	I/O	UART Rx during flash programming; GPIO3
26	U0TXD	I/O	UART TX during flash programming; GPIO1; SPI_CS1
27	XTAL_OUT	I/O	Connect to crystal oscillator output, can be used to provide BT clock input
28	XTAL_IN	I/O	Connect to crystal oscillator input
29	VDDD	P	Analog Power 2.5 V ~ 3.6 V
30	VDDA	P	Analog Power 2.5 V ~ 3.6 V
31	RES12K	I	Serial connection with a 12 kΩ resistor and connect to the ground
32	EXT_RSTB	I	External reset signal (Low voltage level: active)

📖 ***Note:***

1. *GPIO2, GPIO0, and MTDO are used to select booting mode and the SDIO mode;*
2. *U0TXD should not be pulled externally to a low logic level during the powering-up.*

3. Functional Description

The functional diagram of ESP8266EX is shown as in Figure 3-1.

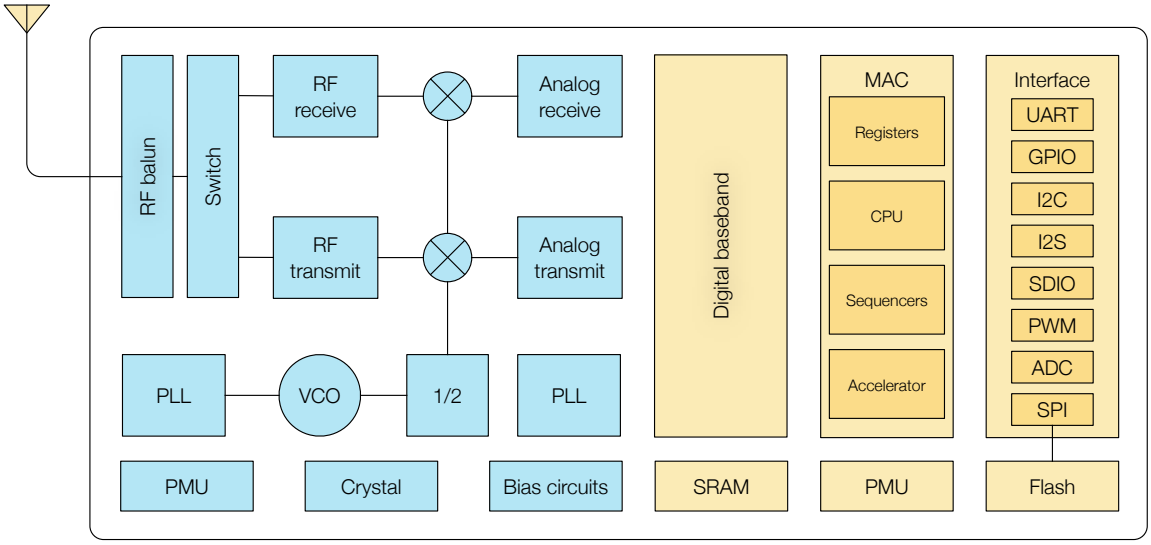


Figure 3-1. Functional Block Diagram

3.1. CPU, Memory, and Flash

3.1.1. CPU

The ESP8266EX integrates a Tensilica L106 32-bit RISC processor, which achieves extra-low power consumption and reaches a maximum clock speed of 160 MHz. The Real-Time Operating System (RTOS) and Wi-Fi stack allow 80% of the processing power to be available for user application programming and development. The CPU includes the interfaces as below:

- Programmable RAM/ROM interfaces (iBus), which can be connected with memory controller, and can also be used to visit flash.
- Data RAM interface (dBus), which can connected with memory controller.
- AHB interface which can be used to visit the register.

3.1.2. Memory

ESP8266EX Wi-Fi SoC integrates memory controller and memory units including SRAM and ROM. MCU can access the memory units through iBus, dBus, and AHB interfaces. All memory units can be accessed upon request, while a memory arbiter will decide the running sequence according to the time when these requests are received by the processor.

According to our current version of SDK, SRAM space available to users is assigned as below.

- RAM size < 50 kB, that is, when ESP8266EX is working under the Station mode and connects to the router, the maximum programmable space accessible in Heap + Data section is around 50 kB.
- There is no programmable ROM in the SoC. Therefore, user program must be stored in an external SPI flash.

3.1.3. External Flash

ESP8266EX uses external SPI flash to store user programs, and supports up to 16 MB memory capacity theoretically.

The minimum flash memory of ESP8266EX is shown below:

- OTA disabled: 512 kB at least
- OTA enabled: 1 MB at least

> ⚠ **Notice:**
>
> *SPI mode supported: Standard SPI, Dual SPI and Quad SPI. The correct SPI mode should be selected when flashing bin files to ESP8266. Otherwise, the downloaded firmware/program may not be working properly.*

3.2. Clock

3.2.1. High Frequency Clock

The high frequency clock on ESP8266EX is used to drive both transmit and receive mixers. This clock is generated from internal crystal oscillator and external crystal. The crystal frequency ranges from 24 MHz to 52 MHz.

The internal calibration inside the crystal oscillator ensures that a wide range of crystals can be used, nevertheless the quality of the crystal is still a factor to consider to have reasonable phase noise and good Wi-Fi sensitivity. Refer to Table 3-1 to measure the frequency offset.

Table 3-1. High Frequency Clock Specifications

Parameter	Symbol	Min	Max	Unit
Frequency	FXO	24	52	MHz
Loading capacitance	CL	-	32	pF
Motional capacitance	CM	2	5	pF
Series resistance	RS	0	65	Ω

Parameter	Symbol	Min	Max	Unit
Frequency tolerance	ΔFXO	–15	15	ppm
Frequency vs temperature (–25 °C ~ 75 °C)	ΔFXO,Temp	–15	15	ppm

3.2.2. External Clock Requirements

An externally generated clock is available with the frequency ranging from 24 MHz to 52 MHz. The following characteristics are expected to achieve good performance of radio.

Table 3-2. External Clock Reference

Parameter	Symbol	Min	Max	Unit
Clock amplitude	VXO	0.8	1.5	Vpp
External clock accuracy	ΔFXO,EXT	–15	15	ppm
Phase noise @1-kHz offset, 40-MHz clock	-	-	–120	dBc/Hz
Phase noise @10-kHz offset, 40-MHz clock	-	-	–130	dBc/Hz
Phase noise @100-kHz offset, 40-MHz clock	-	-	–138	dBc/Hz

3.3. Radio

ESP8266EX radio consists of the following blocks.

- 2.4 GHz receiver
- 2.4 GHz transmitter
- High speed clock generators and crystal oscillator
- Bias and regulators
- Power management

3.3.1. Channel Frequencies

The RF transceiver supports the following channels according to IEEE802.11 b/g/n standards.

Table 3-3. Frequency Channel

Channel No.	Frequency (MHz)	Channel No.	Frequency (MHz)
1	2412	8	2447
2	2417	9	2452
3	2422	10	2457
4	2427	11	2462

Channel No.	Frequency (MHz)	Channel No.	Frequency (MHz)
5	2432	12	2467
6	2437	13	2472
7	2442	14	2484

3.3.2. 2.4 GHz Receiver

The 2.4 GHz receiver down-converts the RF signals to quadrature baseband signals and converts them to the digital domain with 2 high resolution high speed ADCs. To adapt to varying signal channel conditions, RF filters, automatic gain control (AGC), DC offset cancelation circuits and baseband filters are integrated within ESP8266EX.

3.3.3. 2.4 GHz Transmitter

The 2.4 GHz transmitter up-converts the quadrature baseband signals to 2.4 GHz, and drives the antenna with a high-power CMOS power amplifier. The function of digital calibration further improves the linearity of the power amplifier, enabling a state of art performance of delivering +19.5 dBm average TX power for 802.11b transmission and +18 dBm for 802.11n (MSC0) transmission.

Additional calibrations are integrated to offset any imperfections of the radio, such as:

- Carrier leakage
- I/Q phase matching
- Baseband nonlinearities

These built-in calibration functions reduce the product test time and make the test equipment unnecessary.

3.3.4. Clock Generator

The clock generator generates quadrature 2.4 GHz clock signals for the receiver and transmitter. All components of the clock generator are integrated on the chip, including all inductors, varactors, loop filters, linear voltage regulators and dividers.

The clock generator has built-in calibration and self test circuits. Quadrature clock phases and phase noise are optimized on-chip with patented calibration algorithms to ensure the best performance of the receiver and transmitter.

3.4. Wi-Fi

ESP8266EX implements TCP/IP and full 802.11 b/g/n WLAN MAC protocol. It supports Basic Service Set (BSS) STA and SoftAP operations under the Distributed Control Function (DCF). Power management is handled with minimum host interaction to minimize active-duty period.

3.4.1. Wi-Fi Radio and Baseband

The ESP8266EX Wi-Fi Radio and Baseband support the following features:

- 802.11 b and 802.11 g
- 802.11 n MCS0-7 in 20 MHz bandwidth
- 802.11 n 0.4 µs guard-interval
- up to 72.2 Mbps of data rate
- Receiving STBC 2 x 1
- Up to 20.5 dBm of transmitting power
- Adjustable transmitting power

3.4.2. Wi-Fi MAC

The ESP8266EX Wi-Fi MAC applies low-level protocol functions automatically, as follows:

- 2 × virtual Wi-Fi interfaces
- Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode
- Request To Send (RTS), Clear To Send (CTS) and Immediate Block ACK
- Defragmentation
- CCMP (CBC-MAC, counter mode), TKIP (MIC, RC4), WEP (RC4) and CRC
- Automatic beacon monitoring (hardware TSF)
- Dual and single antenna Bluetooth co-existence support with optional simultaneous receive (Wi-Fi/Bluetooth) capability

3.5. Power Management

ESP8266EX is designed with advanced power management technologies and intended for mobile devices, wearable electronics and the Internet of Things applications.

The low-power architecture operates in the following modes:

- Active mode: The chip radio is powered on. The chip can receive, transmit, or listen.
- Modem-sleep mode: The CPU is operational. The Wi-Fi and radio are disabled.
- Light-sleep mode: The CPU and all peripherals are paused. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
- Deep-sleep mode: Only the RTC is operational and all other part of the chip are powered off.

Table 3-4. Power Consumption by Power Modes

Power Mode	Description	Power Consumption
Active (RF working)	Wi-Fi TX packet	Please refer to Table 5-2.
	Wi-Fi RX packet	
Modem-sleep①	CPU is working	15 mA
Light-sleep②	-	0.9 mA
Deep-sleep③	Only RTC is working	20 uA
Shut down	-	0.5 uA

📖 ***Notes:***

① ***Modem-sleep*** *mode is used in the applications that require the CPU to be working, as in PWM or I2S applications. According to 802.11 standards (like U-APSD), it shuts down the Wi-Fi Modem circuit while maintaining a Wi-Fi connection with no data transmission to optimize power consumption. E.g., in DTIM3, maintaining a sleep of 300 ms with a wakeup of 3 ms cycle to receive AP's Beacon packages at interval requires about 15 mA current.*

② *During* ***Light-sleep*** *mode, the CPU may be suspended in applications like Wi-Fi switch. Without data transmission, the Wi-Fi Modem circuit can be turned off and CPU suspended to save power consumption according to the 802.11 standards (U-APSD). E.g. in DTIM3, maintaining a sleep of 300 ms with a wakeup of 3ms to receive AP's Beacon packages at interval requires about 0.9 mA current.*

③ *During* ***Deep-sleep*** *mode, Wi-Fi is turned off. For applications with long time lags between data transmission, e.g. a temperature sensor that detects the temperature every 100 s, sleeps for 300 s and wakes up to connect to the AP (taking about 0.3 ~ 1 s), the overall average current is less than 1 mA. The current of 20 μA is acquired at the voltage of 2.5 V.*

4. Peripheral Interface

4.1. General Purpose Input/Output Interface (GPIO)

ESP8266EX has 17 GPIO pins which can be assigned to various functions by programming the appropriate registers.

Each GPIO PAD can be configured with internal pull-up or pull-down (XPD_DCDC can only be configured with internal pull-down, other GPIO PAD can only be configured with internal pull-up), or set to high impedance. When configured as an input, the data are stored in software registers; the input can also be set to edge-trigger or level trigger CPU interrupts. In short, the IO pads are bi-directional, non-inverting and tristate, which includes input and output buffer with tristate control inputs.

These pins, when working as GPIOs, can be multiplexed with other functions such as I2C, I2S, UART, PWM, and IR Remote Control, etc.

4.2. Secure Digital Input/Output Interface (SDIO)

ESP8266EX has one Slave SDIO, the definitions of which are described as Table 4-1, which supports 25 MHz SDIO v1.1 and 50 MHz SDIO v2.0, and 1 bit/4 bit SD mode and SPI mode.

Table 4-1. Pin Definitions of SDIOs

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SDIO_CLK
SDIO_DATA0	22	IO7	SDIO_DATA0
SDIO_DATA1	23	IO8	SDIO_DATA1
SDIO_DATA_2	18	IO9	SDIO_DATA_2
SDIO_DATA_3	19	IO10	SDIO_DATA_3
SDIO_CMD	20	IO11	SDIO_CMD

4.3. Serial Peripheral Interface (SPI/HSPI)

ESP8266EX has two SPIs.

- One general Slave/Master SPI
- One general Slave/Master HSPI

Functions of all these pins can be implemented via hardware.

4.3.1. General SPI (Master/Slave)

Table 4-2. Pin Definitions of SPIs

Pin Name	Pin Num	IO	Function Name
SDIO_CLK	21	IO6	SPICLK
SDIO_DATA0	22	IO7	SPIQ/MISO
SDIO_DATA1	23	IO8	SPID/MOSI
SDIO_DATA_2	18	IO9	SPIHD
SDIO_DATA_3	19	IO10	SPIWP
U0TXD	26	IO1	SPICS1
GPIO0	15	IO0	SPICS2
SDIO_CMD	20	IO11	SPICS0

📖 ***Note:***

SPI mode can be implemented via software programming. The clock frequency is 80 MHz at maximum when working as a master, 20 MHz at maximum when working as a slave.

4.3.2. HSPI (Master/Slave)

Table 4-3. Pin Definitions of HSPI

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	HSPICLK
MTDI	10	IO12	HSPIQ/MISO
MTCK	12	IO13	HSPID/MOSI
MTDO	13	IO15	HPSICS

📖 ***Note:***

SPI mode can be implemented via software programming. The clock frequency is 20 MHz at maximum.

4.4. I2C Interface

ESP8266EX has one I2C, which is realized via software programming, used to connect with other microcontrollers and other peripheral equipments such as sensors. The pin definition of I2C is as below.

Table 4-4. Pin Definitions of I2C

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	I2C_SCL
GPIO2	14	IO2	I2C_SDA

Both I2C Master and I2C Slave are supported. I2C interface functionality can be realized via software programming, and the clock frequency is 100 kHz at maximum.

4.5. I2S Interface

ESP8266EX has one I2S data input interface and one I2S data output interface, and supports the linked list DMA. I2S interfaces are mainly used in applications such as data collection, processing, and transmission of audio data, as well as the input and output of serial data. For example, LED lights (WS2812 series) are supported. The pin definition of I2S is shown in Table 4-5.

Table 4-5. Pin Definitions of I2S

I2S Data Input			
Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	I2SI_DATA
MTCK	12	IO13	I2SI_BCK
MTMS	9	IO14	I2SI_WS
MTDO	13	IO15	I2SO_BCK
U0RXD	25	IO3	I2SO_DATA
GPIO2	14	IO2	I2SO_WS

4.6. Universal Asynchronous Receiver Transmitter (UART)

ESP8266EX has two UART interfaces UART0 and UART1, the definitions are shown in Table 4-6.

Table 4-6. Pin Definitions of UART

Pin Type	Pin Name	Pin Num	IO	Function Name
UART0	U0RXD	25	IO3	U0RXD
	U0TXD	26	IO1	U0TXD
	MTDO	13	IO15	U0RTS
	MTCK	12	IO13	U0CTS
UART1	GPIO2	14	IO2	U1TXD
	SD_D1	23	IO8	U1RXD

Data transfers to/from UART interfaces can be implemented via hardware. The data transmission speed via UART interfaces reaches 115200 x 40 (4.5 Mbps).

UART0 can be used for communication. It supports flow control. Since UART1 features only data transmit signal (TX), it is usually used for printing log.

> 📖 ***Note:***
>
> *By default, UART0 outputs some printed information when the device is powered on and booting up. The baud rate of the printed information is relevant to the frequency of the external crystal oscillator. If the frequency of the crystal oscillator is 40 MHz, then the baud rate for printing is 115200; if the frequency of the crystal oscillator is 26 MHz, then the baud rate for printing is 74880. If the printed information exerts any influence on the functionality of the device, it is suggested to block the printing during the power-on period by changing (U0TXD, U0RXD) to (MTDO, MTCK).*

4.7. Pulse-Width Modulation (PWM)

ESP8266EX has four PWM output interfaces. They can be extended by users themselves. The pin definitions of the PWM interfaces are defined as below.

Table 4-7. Pin Definitions of PWM

Pin Name	Pin Num	IO	Function Name
MTDI	10	IO12	PWM0
MTDO	13	IO15	PWM1
MTMS	9	IO14	PWM2
GPIO4	16	IO4	PWM3

The functionality of PWM interfaces can be implemented via software programming. For example, in the LED smart light demo, the function of PWM is realized by interruption of the timer, the minimum resolution reaches as high as 44 ns. PWM frequency range is adjustable from 1000 μs to 10000 μs, i.e., between 100 Hz and 1 kHz. When the PWM

frequency is 1 kHz, the duty ratio will be 1/22727, and a resolution of over 14 bits will be achieved at 1 kHz refresh rate.

4.8. IR Remote Control

ESP8266EX currently supports one infrared remote control interface. For detailed pin definitions, please see Table 4-8 below.

Table 4-8. Pin Definitions of IR Remote Control

Pin Name	Pin Num	IO	Function Name
MTMS	9	IO14	IR TX
GPIO5	24	IO 5	IR Rx

The functionality of Infrared remote control interface can be implemented via software programming. NEC coding, modulation, and demodulation are supported by this interface. The frequency of modulated carrier signal is 38 kHz, while the duty ratio of the square wave is 1/3. The transmission range is around 1m which is determined by two factors: one is the maximum current drive output, the other is internal current-limiting resistance value in the infrared receiver. The larger the resistance value, the lower the current, so is the power, and vice versa.

4.9. ADC (Analog-to-Digital Converter)

ESP8266EX is embedded with a 10-bit precision SAR ADC. TOUT (Pin6) is defined as below:

Table 4-9. Pin Definition of ADC

Pin Name	Pin Num	Function Name
TOUT	6	ADC Interface

The following two measurements can be implemented using ADC (Pin6). However, they cannot be implemented at the same time.

- Measure the power supply voltage of VDD3P3 (Pin3 and Pin4).

Hardware Design	TOUT must be floating.
RF Initialization Parameter	The 107th byte of ***esp_init_data_default.bin*** (0 ~ 127 bytes), `vdd33_const` must be set to `0xFF`.
RF Calibration Process	Optimize the RF circuit conditions based on the testing results of VDD3P3 (Pin3 and Pin4).
User Programming	Use `system_get_vdd33` instead of `system_adc_read`.

- Measure the input voltage of TOUT (Pin6).

Hardware Design	The input voltage range is 0 to 1.0 V when TOUT is connected to external circuit.
RF Initialization Parameter	The value of the 107th byte of ***esp_init_data_default.bin*** (0 ~ 127 bytes), `vdd33_const` must be set to the real power supply voltage of Pin3 and Pin4. The unit and effective value range of `vdd33_const` is 0.1 V and 18 to 36, respectively, thus making the working power voltage range of ESP8266EX between 1.8 V and 3.6 V.
RF Calibration Process	Optimize the RF circuit conditions based on the value of `vdd33_const`. The permissible error is ± 0.2 V.
User Programming	Use `system_adc_read` instead of `system_get_vdd33`.

📖 ***Notes:***

esp_init_data_default.bin *is provided in SDK package which contains RF initialization parameters (0 ~ 127 bytes). The name of the 107th byte in* ***esp_init_data_default.bin*** *is* `vdd33_const`, *which is defined as below:*

- *When vdd33_const = 0xff, the power voltage of Pin3 and Pin4 will be tested by the internal self-calibration process of ESP8266EX itself. RF circuit conditions should be optimized according to the testing results.*
- *When 18 = < vdd33_const = < 36, ESP8266EX RF Calibration and optimization process is implemented via (vdd33_const/10).*
- *When vdd33_const < 18 or 36 < vdd33_const < 255, vdd33_const is invalid. ESP8266EX RF Calibration and optimization process is implemented via the default value 3.3 V.*

5. Electrical Specifications

5.1. Electrical Characteristics

Table 5-1. Electrical Characteristics

Parameters		Conditions	Min	Typical	Max	Unit
Operating Temperature Range		-	–40	Normal	125	°C
Maximum Soldering Temperature		IPC/JEDEC J-STD-020	-	-	260	°C
Working Voltage Value		-	2.5	3.3	3.6	V
I/O	V_{IL}	-	–0.3	-	0.25 V_{IO}	V
	V_{IH}		0.75 V_{IO}		3.6	
	V_{OL}	-	-	-	0.1 V_{IO}	
	V_{OH}		0.8 V_{IO}		-	
	I_{MAX}	-	-	-	12	mA
Electrostatic Discharge (HBM)		TAMB = 25 °C	-	-	2	KV
Electrostatic Discharge (CDM)		TAMB = 25 °C	-	-	0.5	KV

Notes on CHIP_EN:

The figure below shows ESP8266EX power-up and reset timing. Details about the parameters are listed in Table 5-2.

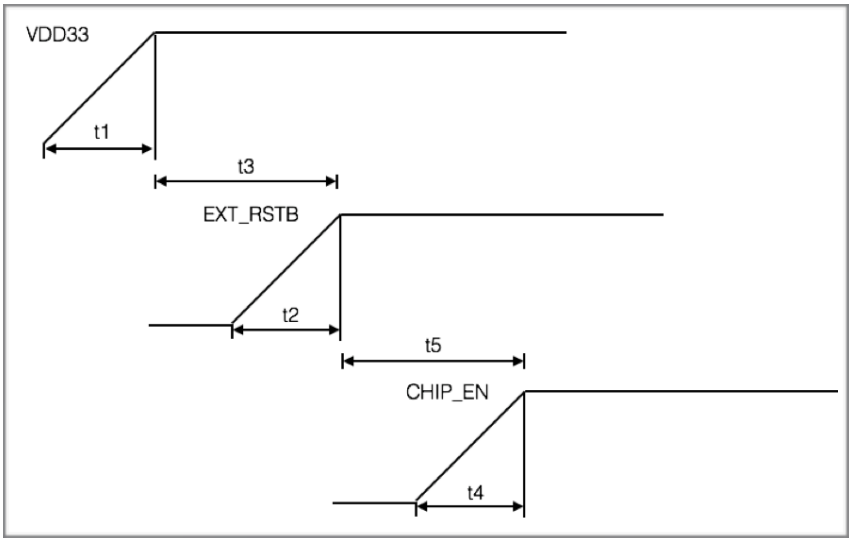


Figure 5-1. ESP8266EX Power-up and Reset Timing

Table 5-2. Description of ESP8266EX Power-up and Reset Timing Parameters

	Description	Min	Max	Unit
t1	The rise-time of VDD33	10	2000	µs
t2	The rise-time of EXT_RSTB	0	2	ms
t3	EXT_RSTB goes high after VDD33	0.1	-	ms
t4	The rise-time of CHIP_EN	0	2	ms
t5	CHIP_EN goes high after EXT_RSTB	0.1	-	ms

5.2. RF Power Consumption

Unless otherwise specified, the power consumption measurements are taken with a 3.0 V supply at 25 °C of ambient temperature. All transmitters' measurements are based on a 50% duty cycle.

Table 5-3. Power Consumption

Parameters	Min	Typical	Max	Unit
TX802.11 b, CCK 11 Mbps, P_{OUT} = +17 dBm	-	170	-	mA
TX802.11 g, OFDM 54Mbps, P_{OUT} = +15 dBm	-	140	-	mA
TX802.11 n, MCS7, P_{OUT} = +13 dBm	-	120	-	mA
Rx802.11 b, 1024 bytes packet length, –80 dBm	-	50	-	mA
Rx802.11 g, 1024 bytes packet length, –70 dBm	-	56	-	mA
Rx802.11 n, 1024 bytes packet length, –65 dBm	-	56	-	mA

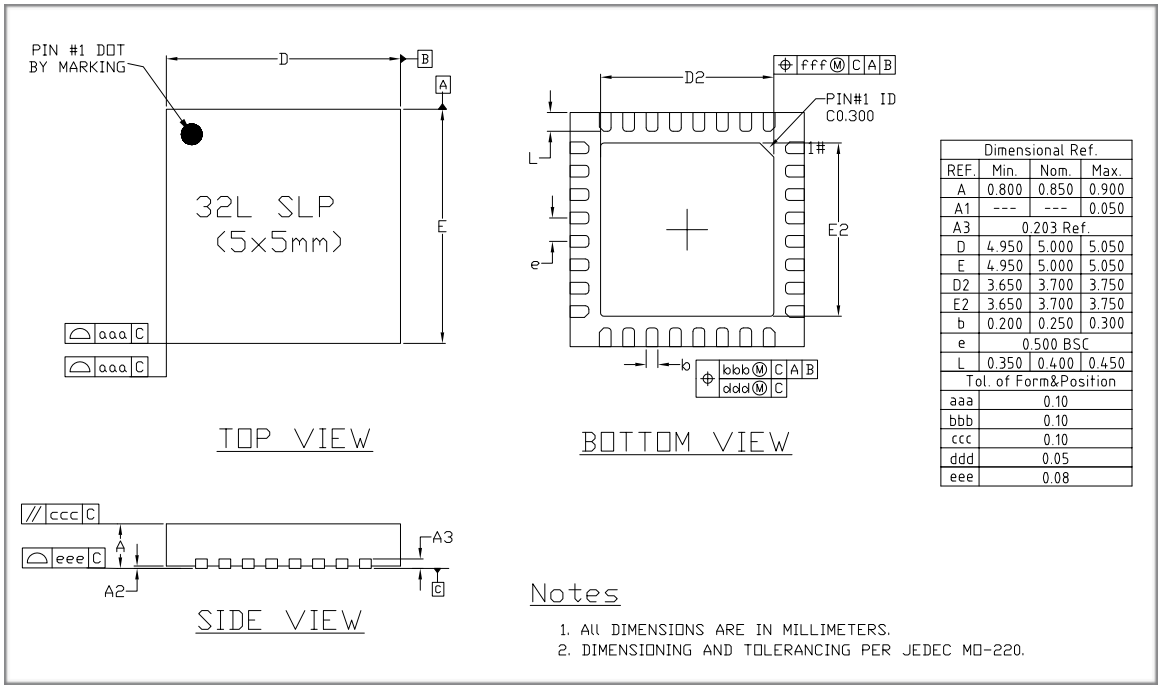
5.3. Wi-Fi Radio Characteristics

The following data are from tests conducted at room temperature, with a 3.3 V power supply.

Table 5-3. Wi-Fi Radio Characteristics

Parameters	Min	Typical	Max	Unit
Input frequency	2412	-	2484	MHz
Output impedance	-	39 + j6	-	Ω
Output power of PA for 72.2 Mbps	15.5	16.5	17.5	dBm
Output power of PA for 11b mode	19.5	20.5	21.5	dBm
Sensitivity				
DSSS, 1 Mbps	-	–98	-	dBm
CCK, 11 Mbps	-	–91	-	dBm
6 Mbps (1/2 BPSK)	-	–93	-	dBm
54 Mbps (3/4 64-QAM)	-	–75	-	dBm
HT20, MCS7 (65 Mbps, 72.2 Mbps)	-	–72	-	dBm
Adjacent Channel Rejection				
OFDM, 6 Mbps	-	37	-	dB
OFDM, 54 Mbps	-	21	-	dB
HT20, MCS0	-	37	-	dB
HT20, MCS7	-	20	-	dB

6. Package Information



Dimensional Ref.			
REF.	Min.	Nom.	Max.
A	0.800	0.850	0.900
A1	---	---	0.050
A3	0.203 Ref.		
D	4.950	5.000	5.050
E	4.950	5.000	5.050
D2	3.650	3.700	3.750
E2	3.650	3.700	3.750
b	0.200	0.250	0.300
e	0.500 BSC		
L	0.350	0.400	0.450
Tol. of Form&Position			
aaa	0.10		
bbb	0.10		
ccc	0.10		
ddd	0.05		
eee	0.08		

Figure 6-1. ESP8266EX Package

I. Appendix - Pin List

For detailed pin information, please see *ESP8266 Pin List*.

- Digital Die Pin List
- Buffer Sheet
- Register List
- Strapping List

> 📖 ***Notes:***
>
> - *INST_NAME refers to the IO_MUX REGISTER defined in* ***eagle_soc.h****, for example MTDI_U refers to PERIPHS_IO_MUX_MTDI_U.*
> - *Net Name refers to the pin name in schematic.*
> - *Function refers to the multifunction of each pin pad.*
> - *Function number 1 ~ 5 correspond to FUNCTION 0 ~ 4 in SDK. For example, set MTDI to GPIO12 as follows.*
>   - `#define FUNC_GPIO12  3 //defined in eagle_soc.h`
>   - `PIN_FUNC_SELECT(PERIPHS_IO_MUX_MTDI_U,FUNC_GPIO12)`

II. Appendix - Learning Resources

II.1. Must-Read Documents

- *ESP8266 Quick Start Guide*

  Description: This document is a quick user guide to getting started with ESP8266. It includes an introduction to the ESP-LAUNCHER, instructions on how to download firmware to the board and run it, how to compile the AT application, as well as the structure and debugging method of RTOS SDK. Basic documentation and other related resources for the ESP8266 are also provided.
- *ESP8266 SDK Getting Started Guide*

  Description: This document takes ESP-LAUNCHER and ESP-WROOM-02 as examples of how to use the ESP8266 SDK. The contents include preparations before compilation, SDK compilation and firmware download.
- *ESP8266 Pin List*

  Description: This link directs you to a list containing the type and function of every ESP8266 pin.
- *ESP8266 Hardware Design Guideline*

  Description: This document provides a technical description of the ESP8266 series of products, including ESP8266EX, ESP-LAUNCHER and ESP-WROOM.
- *ESP8266 Technical Reference*

  Description: This document provides an introduction to the interfaces integrated on ESP8266. Functional overview, parameter configuration, function description, application demos and other pieces of information are included.
- *ESP8266 Hardware Resources*

  Description: This zip package includes manufacturing BOMs, schematics and PCB layouts of ESP8266 boards and modules.
- *FAQ*

II.2. Must-Have Resources

- ESP8266 SDKs

  Description: This webpage provides links both to the latest version of the ESP8266 SDK and the older ones.
- ESP8266 Tools

Description: This webpage provides links to both the ESP8266 flash download tools and the ESP8266 performance evaluation tools.

- ESP8266 Apps
- ESP8266 Certification and Test Guide
- ESP8266 BBS
- ESP8266 Resources

Espressif IOT Team

www.espressif.com

APPENDIX B: broadcom bcM2835 PERIPHERALS

BCM2835 ARM Peripherals

Broadcom Europe Ltd. 406 Science Park Milton Road Cambridge CB4 0WW

Table of Contents

1 Introduction

1.1 Overview

BCM2835 contains the following peripherals which may safely be accessed by the ARM:

- Timers
- Interrupt controller
- GPIO
- USB
- PCM / I2S
- DMA controller
- I2C master
- I2C / SPI slave
- SPI0, SPI1, SPI2
- PWM
- UART0, UART1

The purpose of this datasheet is to provide documentation for these peripherals in sufficient detail to allow a developer to port an operating system to BCM2835.

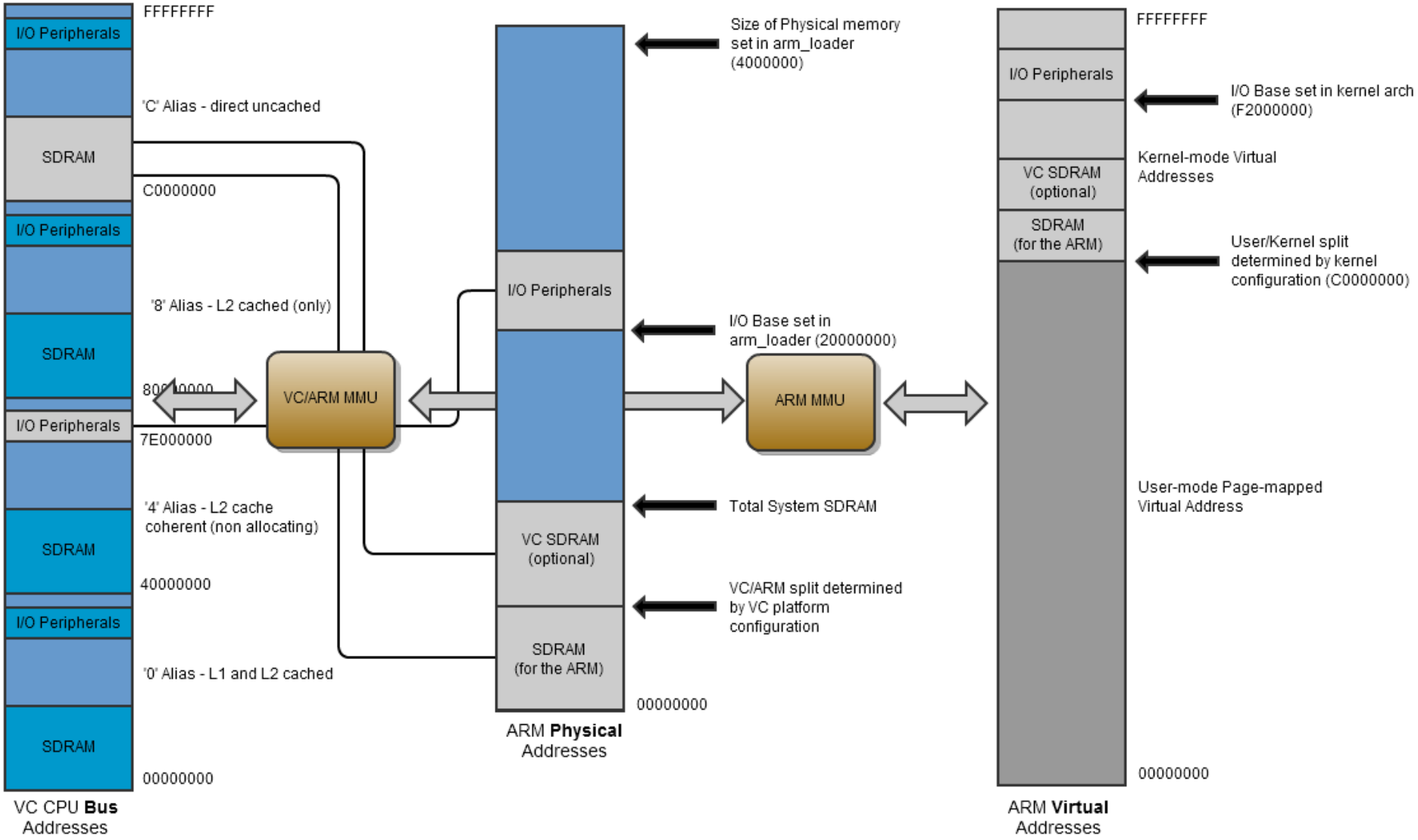
There are a number of peripherals which are intended to be controlled by the GPU. These are omitted from this datasheet. Accessing these peripherals from the ARM is not recommended.

1.2 Address map

1.2.1 Diagrammatic overview

In addition to the ARM's MMU, BCM2835 includes a second coarse-grained MMU for mapping ARM physical addresses onto system bus addresses. This diagram shows the main address spaces of interest:

BCM2835 ARM Peripherals



Addresses in ARM Linux are:

- issued as virtual addresses by the ARM core, then
- mapped into a physical address by the ARM MMU, then
- mapped into a bus address by the ARM mapping MMU, and finally
- used to select the appropriate peripheral or location in RAM.

1.2.2 ARM virtual addresses (standard Linux kernel only)

As is standard practice, the standard BCM2835 Linux kernel provides a contiguous mapping over the whole of available RAM at the top of memory. The kernel is configured for a 1GB/3GB split between kernel and user-space memory.

The split between ARM and GPU memory is selected by installing one of the supplied start*.elf files as start.elf in the FAT32 boot partition of the SD card. The minimum amount of memory which can be given to the GPU is 32MB, but that will restrict the multimedia performance; for example, 32MB does not provide enough buffering for the GPU to do 1080p30 video decoding.

Virtual addresses in kernel mode will range between 0xC0000000 and 0xEFFFFFFF.

Virtual addresses in user mode (i.e. seen by processes running in ARM Linux) will range between 0x00000000 and 0xBFFFFFFF.

Peripherals (at physical address 0x20000000 on) are mapped into the kernel virtual address space starting at address 0xF2000000. Thus a peripheral advertised here at bus address 0x7Ennnnnn is available in the ARM kenel at virtual address 0xF2nnnnnn.

1.2.3 ARM physical addresses

Physical addresses start at 0x00000000 for RAM.

- The ARM section of the RAM starts at 0x00000000.
- The VideoCore section of the RAM is mapped in only if the system is configured to support a memory mapped display (this is the common case).

The VideoCore MMU maps the ARM physical address space to the bus address space seen by VideoCore (and VideoCore peripherals). The bus addresses for RAM are set up to map onto the uncached[1] bus address range on the VideoCore starting at 0xC0000000.

Physical addresses range from 0x20000000 to 0x20FFFFFF for peripherals. The bus addresses for peripherals are set up to map onto the peripheral bus address range starting at 0x7E000000. Thus a peripheral advertised here at bus address 0x7Ennnnnn is available at physical address 0x20nnnnnn.

1.2.4 Bus addresses

The peripheral addresses specified in this document are bus addresses. Software directly accessing peripherals must translate these addresses into physical or virtual addresses, as described above. Software accessing peripherals using the DMA engines must use bus addresses.

[1] BCM2835 provides a 128KB system L2 cache, which is used primarily by the GPU. Accesses to memory are routed either via or around the L2 cache depending on senior two bits of the bus address.

Software accessing RAM directly must use physical addresses (based at 0x00000000). Software accessing RAM using the DMA engines must use bus addresses (based at 0xC0000000).

1.3 Peripheral access precautions for correct memory ordering

The BCM2835 system uses an AMBA AXI-compatible interface structure. In order to keep the system complexity low and data throughput high, the BCM2835 AXI system does not always return read data in-order[2]. The GPU has special logic to cope with data arriving out-of-order; however the ARM core does not contain such logic. Therefore some precautions must be taken when using the ARM to access peripherals.

Accesses to the same peripheral will always arrive and return in-order. It is only when switching from one peripheral to another that data can arrive out-of-order. The simplest way to make sure that data is processed in-order is to place a memory barrier instruction at critical positions in the code. You should place:

- A memory write barrier before the first write to a peripheral.
- A memory read barrier after the last read of a peripheral.

It is **not** required to put a memory barrier instruction after **each** read or write access. Only at those places in the code where it is possible that a peripheral read or write may be followed by a read or write of a **different** peripheral. This is normally at the entry and exit points of the peripheral service code.

As interrupts can appear anywhere in the code so you should safeguard those. If an interrupt routine reads from a peripheral the routine should start with a memory read barrier. If an interrupt routine writes to a peripheral the routine should end with a memory write barrier.

[2]Normally a processor assumes that if it executes two read operations the data will arrive in order. So a read from location X followed by a read from location Y should return the data of location X first, followed by the data of location Y. Data arriving out of order can have disastrous consequences. For example:

```
a_status = *pointer_to_peripheral_a;
b_status = *pointer_to_peripheral_b;
```

Without precuations the values ending up in the variables a_status and b_status can be swapped around.

It is theoretical possible for writes to go 'wrong' but that is far more difficult to achieve. The AXI system makes sure the data always arrives in-order at its intended destination. So:

```
*pointer_to_peripheral_a = value_a;
*pointer_to_peripheral_b = value_b;
```

will always give the expected result. The only time write data can arrive out-of-order is if two different peripherals are connected to the same external equipment.

2 Auxiliaries: UART1 & SPI1, SPI2

2.1 Overview

The Device has three Auxiliary peripherals: One mini UART and two SPI masters. These three peripheral are grouped together as they share the same area in the peripheral register map and they share a common interrupt. Also all three are controlled by the auxiliary enable register.

Auxiliary peripherals Register Map (offset = 0x7E21 5000)			
Address	**Register Name[3]**	**Description**	**Size**
0x7E21 5000	AUX_IRQ	Auxiliary Interrupt status	3
0x7E21 5004	AUX_ENABLES	Auxiliary enables	3
0x7E21 5040	AUX_MU_IO_REG	Mini Uart I/O Data	8
0x7E21 5044	AUX_MU_IER_REG	Mini Uart Interrupt Enable	8
0x7E21 5048	AUX_MU_IIR_REG	Mini Uart Interrupt Identify	8
0x7E21 504C	AUX_MU_LCR_REG	Mini Uart Line Control	8
0x7E21 5050	AUX_MU_MCR_REG	Mini Uart Modem Control	8
0x7E21 5054	AUX_MU_LSR_REG	Mini Uart Line Status	8
0x7E21 5058	AUX_MU_MSR_REG	Mini Uart Modem Status	8
0x7E21 505C	AUX_MU_SCRATCH	Mini Uart Scratch	8
0x7E21 5060	AUX_MU_CNTL_REG	Mini Uart Extra Control	8
0x7E21 5064	AUX_MU_STAT_REG	Mini Uart Extra Status	32
0x7E21 5068	AUX_MU_BAUD_REG	Mini Uart Baudrate	16
0x7E21 5080	AUX_SPI0_CNTL0_REG	SPI 1 Control register 0	32
0x7E21 5084	AUX_SPI0_CNTL1_REG	SPI 1 Control register 1	8
0x7E21 5088	AUX_SPI0_STAT_REG	SPI 1 Status	32
0x7E21 5090	AUX_SPI0_IO_REG	SPI 1 Data	32
0x7E21 5094	AUX_SPI0_PEEK_REG	SPI 1 Peek	16
0x7E21 50C0	AUX_SPI1_CNTL0_REG	SPI 2 Control register 0	32
0x7E21 50C4	AUX_SPI1_CNTL1_REG	SPI 2 Control register 1	8

[3] These register names are identical to the defines in the AUX_IO header file. For programming purposes these names should be used wherever possible.

0x7E21 50C8	AUX_SPI1_STAT_REG	SPI 2 Status	32
0x7E21 50D0	AUX_SPI1_IO_REG	SPI 2 Data	32
0x7E21 50D4	AUX_SPI1_PEEK_REG	SPI 2 Peek	16

2.1.1 AUX registers

There are two Auxiliary registers which control all three devices. One is the interrupt status register, the second is the Auxiliary enable register. The Auxiliary IRQ status register can help to hierarchically determine the source of an interrupt.

AUXIRQ Register (0x7E21 5000)

SYNOPSIS The AUXIRQ register is used to check any pending interrupts which may be asserted by the three Auxiliary sub blocks.

Bit(s)	Field Name	Description	Type	Reset
31:3		Reserved, write zero, read as don't care		
2	SPI 2 IRQ	If set the SPI 2 module has an interrupt pending.	R	0
1	SPI 1 IRQ	If set the SPI1 module has an interrupt pending.	R	0
0	Mini UART IRQ	If set the mini UART has an interrupt pending.	R	0

AUXENB Register (0x7E21 5004)

SYNOPSIS The AUXENB register is used to enable the three modules; UART, SPI1, SPI2.

Bit(s)	Field Name	Description	Type	Reset
31:3		Reserved, write zero, read as don't care		
2	SPI2 enable	If set the SPI 2 module is enabled. If clear the SPI 2 module is disabled. That also disables any SPI 2 module register access	R/W	0
1	SPI 1 enable	If set the SPI 1 module is enabled. If clear the SPI 1 module is disabled. That also disables any SPI 1 module register access	R/W	0
0	Mini UART enable	If set the mini UART is enabled. The UART will immediately start receiving data, especially if the UART1_RX line is *low*. If clear the mini UART is disabled. That also disables any mini UART register access	R/W	0

If the enable bits are clear you will have ***no access*** to a peripheral. You can not even read or write the registers!

GPIO pins should be set up first the before enabling the UART. The UART core is build to emulate 16550 behaviour. So when it is enabled any data at the inputs will immediately be received . If the UART1_RX line is low (because the GPIO pins have not been set-up yet) that will be seen as a start bit and the UART will start receiving 0x00-characters.

Valid stops bits are not required for the UART. (See also Implementation details). Hence any bit status is acceptable as stop bit and is only used so there is clean timing start for the next bit.

Looking after a reset: the baudrate will be zero and the system clock will be 250 MHz. So only 2.5 µseconds suffice to fill the receive FIFO. The result will be that the FIFO is full and overflowing in no time flat.

2.2 Mini UART

The mini UART is a secondary low throughput[4] UART intended to be used as a console. It needs to be enabled before it can be used. It is also recommended that the correct GPIO function mode is selected before enabling the mini Uart.

The mini Uart has the following features:

- 7 or 8 bit operation.
- 1 start and 1 stop bit.
- No parities.
- Break generation.
- 8 symbols deep FIFOs for receive and transmit.
- SW controlled RTS, SW readable CTS.
- Auto flow control with programmable FIFO level.
- 16550 *like* registers.
- Baudrate derived from system clock.

This is a *mini* UART and it does NOT have the following capabilities:

- Break detection
- Framing errors detection.
- Parity bit
- Receive Time-out interrupt
- DCD, DSR, DTR or RI signals.

The implemented UART is *not* a 16650 compatible UART However as far as possible the first 8 control and status registers are laid out *like* a 16550 UART. Al 16550 register bits which are not supported can be written but will be ignored and read back as 0. All control bits for simple UART receive/transmit operations are available.

[4] The UART itself has no throughput limitations in fact it can run up to 32 Mega baud. But doing so requires significant CPU involvement as it has shallow FIFOs and no DMA support.

2.2.1 Mini UART implementation details.

The UART1_CTS and UART1_RX inputs are synchronised and will take 2 system clock cycles before they are processed.

The module does not check for any framing errors. After receiving a start bit and 8 (or 7) data bits the receiver waits for one half bit time and then starts scanning for the next start bit. The mini UART does *not* check if the stop bit is high or wait for the stop bit to appear. As a result of this a UART1_RX input line which is continuously low (a break condition or an error in connection or GPIO setup) causes the receiver to continuously receive 0x00 symbols.

The mini UART uses 8-times oversampling. The Baudrate can be calculated from:

$$baudrate = \frac{system\_clock\_freq}{8 * (baudrate\_reg + 1)}$$

If the system clock is 250 MHz and the baud register is zero the baudrate is 31.25 Mega baud. (25 Mbits/sec or 3.125 Mbytes/sec). The lowest baudrate with a 250 MHz system clock is 476 Baud.

When writing to the data register only the LS 8 bits are taken. All other bits are ignored. When reading from the data register only the LS 8 bits are valid. All other bits are zero.

2.2.2 Mini UART register details.

AUX_MU_IO_REG Register (0x7E21 5040)

SYNOPSIS The AUX_MU_IO_REG register is primary used to write data to and read data from the UART FIFOs.
If the DLAB bit in the line control register is set this register gives access to the LS 8 bits of the baud rate. (Note: there is easier access to the baud rate register)

Bit(s)	Field Name	Description	Type	Reset
31:8		Reserved, write zero, read as don't care		
7:0	LS 8 bits Baudrate read/write, DLAB=1	Access to the LS 8 bits of the 16-bit baudrate register. (Only If bit 7 of the line control register (DLAB bit) is set)	R/W	0
7:0	Transmit data write, DLAB=0	Data written is put in the transmit FIFO (Provided it is not full) (Only If bit 7 of the line control register (DLAB bit) is clear)	W	0
7:0	Receive data read, DLAB=0	Data read is taken from the receive FIFO (Provided it is not empty) (Only If bit 7 of the line control register (DLAB bit) is clear)	R	0

AUX_MU_IIR_REG Register (0x7E21 5044)

SYNOPSIS The AUX_MU_IER_REG register is primary used to enable interrupts
If the DLAB bit in the line control register is set this register gives access to the MS 8 bits of the baud rate. (Note: there is easier access to the baud rate register)

Bit(s)	Field Name	Description	Type	Reset
31:8		Reserved, write zero, read as don't care		
7:0	MS 8 bits Baudrate read/write, DLAB=1	Access to the MS 8 bits of the 16-bit baudrate register. (Only If bit 7 of the line control register (DLAB bit) is set)	R/w	0
7:2		Reserved, write zero, read as don't care *Some of these bits have functions in a 16550 compatible UART but are ignored here*		
1	Enable receive interrupt (DLAB=0)	If this bit is set the interrupt line is asserted whenever the receive FIFO holds at least 1 byte. If this bit is clear no receive interrupts are generated.	R	0
0	Enable transmit interrupt (DLAB=0)	If this bit is set the interrupt line is asserted whenever the transmit FIFO is empty. If this bit is clear no transmit interrupts are generated.	R	0

AUX_MU_IER_REG Register (0x7E21 5048)

SYNOPSIS The AUX_MU_IIR_REG register shows the interrupt status.
It also has two FIFO enable status bits and (when writing) FIFO clear bits.

Bit(s)	Field Name	Description	Type	Reset
31:8		Reserved, write zero, read as don't care		
7:6	FIFO enables	Both bits always read as 1 as the FIFOs are always enabled	R	11
5:4	-	Always read as zero	R	00
3	-	Always read as zero as the mini UART has no timeout function	R	0
2:1	READ: Interrupt ID bits WRITE: FIFO clear bits	On read this register shows the interrupt ID bit 00 : No interrupts 01 : Transmit holding register empty 10 : Receiver holds valid byte 11 : <Not possible> On write: Writing with bit 1 set will clear the receive FIFO Writing with bit 2 set will clear the transmit FIFO	R/W	00
0	Interrupt pending	This bit is clear whenever an interrupt is pending	R	1

AUX_MU_LCR_REG Register (0x7E21 504C)

SYNOPSIS The AUX_MU_LCR_REG register controls the line data format and gives access to the baudrate register

Bit(s)	Field Name	Description	Type	Reset
31:8		Reserved, write zero, read as don't care		
7	DLAB access	If set the first to Mini UART register give access the the Baudrate register. During operation this bit must be cleared.	R/W	0
6	Break	If set high the UART1_TX line is pulled low continuously. If held for at least 12 bits times that will indicate a break condition.	R/W	0
5:1		Reserved, write zero, read as don't care *Some of these bits have functions in a 16550 compatible UART but are ignored here*		0
0	data size	If clear the UART works in 7-bit mode If set the UART works in 8-bit mode	R/W	0

AUX_MU_MCR_REG Register (0x7E21 5050)

SYNOPSIS The AUX_MU_MCR_REG register controls the 'modem' signals.

Bit(s)	Field Name	Description	Type	Reset
31:8		Reserved, write zero, read as don't care		
7:2		Reserved, write zero, read as don't care *Some of these bits have functions in a 16550 compatible UART but are ignored here*		0
1	RTS	If clear the UART1_RTS line is high If set the UART1_RTS line is low This bit is ignored if the RTS is used for auto-flow control. See the Mini Uart Extra Control register description)	R/W	0
0		Reserved, write zero, read as don't care *This bit has a function in a 16550 compatible UART but is ignored here*		0

AUX_MU_LSR_REG Register (0x7E21 5054)

SYNOPSIS The AUX_MU_LSR_REG register shows the data status.

Bit(s)	Field Name	Description	Type	Reset
31:8		Reserved, write zero, read as don't care		
7		Reserved, write zero, read as don't care *This bit has a function in a 16550 compatible UART but is ignored here*		0
6	Transmitter idle	This bit is set if the transmit FIFO is empty and the transmitter is idle. (Finished shifting out the last bit).	R	1
5	Transmitter empty	This bit is set if the transmit FIFO can accept at least one byte.	R	0
4:2		Reserved, write zero, read as don't care *Some of these bits have functions in a 16550 compatible UART but are ignored here*		0
1	Receiver Overrun	This bit is set if there was a receiver overrun. That is: one or more characters arrived whilst the receive FIFO was full. The newly arrived charters have been discarded. This bit is cleared each time this register is read. To do a non-destructive read of this overrun bit use the Mini Uart Extra Status register.	R/C	0
0	Data ready	This bit is set if the receive FIFO holds at least 1 symbol.	R	0

AUX_MU_MSR_REG Register (0x7E21 5058)

SYNOPSIS The AUX_MU_MSR_REG register shows the 'modem' status.

Bit(s)	Field Name	Description	Type	Reset
31:8		Reserved, write zero, read as don't care		
7:6		Reserved, write zero, read as don't care *Some of these bits have functions in a 16550 compatible UART but are ignored here*		0
5	CTS status	This bit is the inverse of the UART1_CTS input Thus : If set the UART1_CTS pin is low If clear the UART1_CTS pin is high	R	1
3:0		Reserved, write zero, read as don't care *Some of these bits have functions in a 16550 compatible UART but are ignored here*		0

AUX_MU_SCRATCH Register (0x7E21 505C)

SYNOPSIS The AUX_MU_SCRATCH is a single byte storage.

Bit(s)	Field Name	Description	Type	Reset
31:8		Reserved, write zero, read as don't care		
7:0	Scratch	One whole byte extra on top of the 134217728 provided by the SDC	R/W	0

AUX_MU_CNTL_REG Register (0x7E21 5060)

SYNOPSIS The AUX_MU_CNTL_REG provides access to some extra useful and nice features not found on a normal 16550 UART .

Bit(s)	Field Name	Description	Type	Reset
31:8		Reserved, write zero, read as don't care		
7	CTS assert level	This bit allows one to invert the CTS auto flow operation polarity. If set the CTS auto flow assert level is low* If clear the CTS auto flow assert level is high*	R/W	0
6	RTS assert level	This bit allows one to invert the RTS auto flow operation polarity. If set the RTS auto flow assert level is low* If clear the RTS auto flow assert level is high*	R/W	0
5:4	RTS AUTO flow level	These two bits specify at what receiver FIFO level the RTS line is de-asserted in auto-flow mode. 00 : De-assert RTS when the receive FIFO has 3 empty spaces left. 01 : De-assert RTS when the receive FIFO has 2 empty spaces left. 10 : De-assert RTS when the receive FIFO has 1 empty space left. 11 : De-assert RTS when the receive FIFO has 4 empty spaces left.	R/W	0
3	Enable transmit Auto flow-control using CTS	If this bit is set the transmitter will stop if the CTS line is de-asserted. If this bit is clear the transmitter will ignore the status of the CTS line	R/W	0

2	Enable receive Auto flow-control using RTS	If this bit is set the RTS line will de-assert if the receive FIFO reaches it 'auto flow' level. In fact the RTS line will behave as an RTR *(Ready To Receive)* line. If this bit is clear the RTS line is controlled by the AUX_MU_MCR_REG register bit 1.	R/W	0
1	Transmitter enable	If this bit is set the mini UART transmitter is enabled. If this bit is clear the mini UART transmitter is disabled	R/W	1
0	Receiver enable	If this bit is set the mini UART receiver is enabled. If this bit is clear the mini UART receiver is disabled	R/W	1

Receiver enable
If this bit is set no new symbols will be accepted by the receiver. Any symbols in progress of reception will be finished.

Transmitter enable
If this bit is set no new symbols will be send the transmitter. Any symbols in progress of transmission will be finished.

Auto flow control
Automatic flow control can be enabled independent for the receiver and the transmitter.

CTS auto flow control impacts the transmitter only. The transmitter will not send out new symbols when the CTS line is de-asserted. Any symbols in progress of transmission when the CTS line becomes de-asserted will be finished.

RTS auto flow control impacts the receiver only. In fact the name RTS for the control line is incorrect and should be RTR (Ready to Receive). The receiver will de-asserted the RTS (RTR) line when its receive FIFO has a number of empty spaces left. Normally 3 empty spaces should be enough.

If looping back a mini UART using full auto flow control the logic is fast enough to allow the RTS auto flow level of '10' (De-assert RTS when the receive FIFO has 1 empty space left).

Auto flow polarity
To offer full flexibility the polarity of the CTS and RTS (RTR) lines can be programmed. This should allow the mini UART to interface with any existing hardware flow control available.

AUX_MU_STAT_REG Register (0x7E21 5064)

SYNOPSIS The AUX_MU_STAT_REG provides a lot of useful information about the internal status of the mini UART not found on a normal 16550 UART.

Bit(s)	Field Name	Description	Type	Reset
31:28		Reserved, write zero, read as don't care		
27:24	Transmit FIFO fill level	These bits shows how many symbols are stored in the transmit FIFO The value is in the range 0-8	R	0
23:20		Reserved, write zero, read as don't care		
19:16	Receive FIFO fill level	These bits shows how many symbols are stored in the receive FIFO The value is in the range 0-8	R	0
15:10		Reserved, write zero, read as don't care		
9	Transmitter done	This bit is set if the transmitter is idle and the transmit FIFO is empty. It is a logic AND of bits 2 and 8	R	1
8	Transmit FIFO is empty	If this bit is set the transmitter FIFO is empty. Thus it can accept 8 symbols.	R	1
7	CTS line	This bit shows the status of the UART1_CTS line.	R	0
6	RTS status	This bit shows the status of the UART1_RTS line.	R	0
5	Transmit FIFO is full	This is the inverse of bit 1	R	0
4	Receiver overrun	This bit is set if there was a receiver overrun. That is: one or more characters arrived whilst the receive FIFO was full. The newly arrived characters have been discarded. This bit is cleared each time the AUX_MU_LSR_REG register is read.	R	0
3	Transmitter is idle	If this bit is set the transmitter is idle. If this bit is clear the transmitter is idle.	R	1
2	Receiver is idle	If this bit is set the receiver is idle. If this bit is clear the receiver is busy. This bit can change unless the receiver is disabled	R	1
1	Space available	If this bit is set the mini UART transmitter FIFO can accept at least one more symbol. If this bit is clear the mini UART transmitter FIFO is full	R	0

0	Symbol available	If this bit is set the mini UART receive FIFO contains at least 1 symbol If this bit is clear the mini UART receiver FIFO is empty	R	0

Receiver is idle
This bit is only useful if the receiver is disabled. The normal use is to disable the receiver. Then check (or wait) until the bit is set. Now you can be sure that no new symbols will arrive. (e.g. now you can change the baudrate...)

Transmitter is idle
This bit tells if the transmitter is idle. Note that the bit will set only for a short time if the transmit FIFO contains data. Normally you want to use bit 9: Transmitter done.

RTS status
This bit is useful only in receive Auto flow-control mode as it shows the status of the RTS line.

AUX_MU_BAUD Register (0x7E21 5068)

SYNOPSIS The AUX_MU_BAUD register allows direct access to the 16-bit wide baudrate counter.

Bit(s)	Field Name	Description	Type	Reset
31:16		Reserved, write zero, read as don't care		
15:0	Baudrate	mini UART baudrate counter	R/W	0

This is the same register as is accessed using the LABD bit and the first two register, but much easier to access.

2.3 Universal SPI Master (2x)

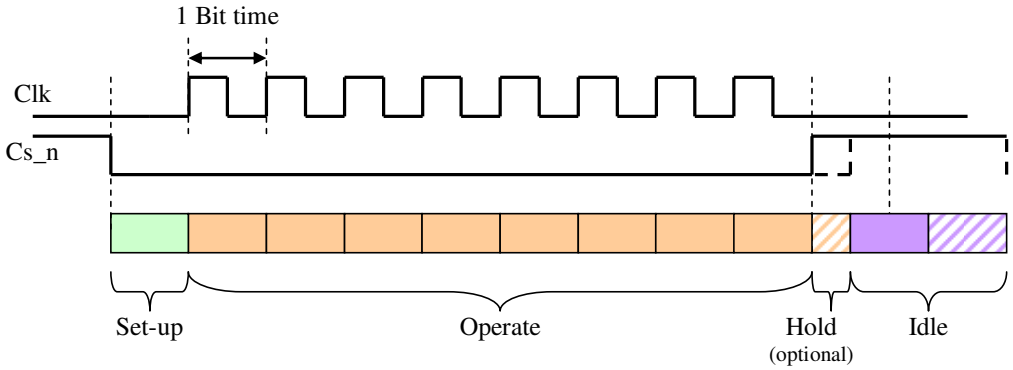
The two universal SPI masters are secondary low throughput[5] SPI interfaces. Like the UART the devices needs to be enabled before they can be used. Each SPI master has the following features:

- Single beat bit length between 1 and 32 bits.
- Single beat variable bit length between 1 and 24 bits
- Multi beat infinite bit length.
- 3 independent chip selects per master.
- 4 entries 32-bit wide transmit and receive FIFOs.
- Data out on rising or falling clock edge.
- Data in on rising or falling clock edge.
- Clock inversion (Idle high or idle low).
- Wide clocking range.
- Programmable data out hold time.
- Shift in/out MS or LS bit first

A major issue with an SPI interface is that there is no SPI standard in any form. Because the SPI interface has been around for a long time some pseudo-standard rules have appeared mostly when interfacing with memory devices. The universal SPI master has been developed to work even with the most 'non-standard' SPI devices.

2.3.1 SPI implementation details

The following diagrams shows a typical SPI access cycle. In this case we have 8 SPI clocks.



One bit time before any clock edge changes the CS_n will go low. This makes sure that the MOSI signal has a full bit-time of set-up against any changing clock edges.

The operation normally ends after the last clock cycle. Note that at the end there is one half-bit time where the clock does not change but which still is part of the operation cycle.

There is an option to add a half bit cycle hold time. This makes sure that any MISO data has at least a full SPI bit time to arrive. (Without this hold time, data clocked out of the SPI device on the last clock edge would have only half a bit time to arrive).

[5] Again the SPIs themselves have no throughput limitations in fact they can run with an SPI clock of 125 MHz. But doing so requires significant CPU involvement as they have shallow FIFOs and no DMA support.

Last there is a guarantee of at least a full bit time where the spi chip select is high. A longer CS_n high period can be programmed for another 1-7 cycles.

The SPI clock frequency is:

$$SPIx\_CLK = \frac{system\_clock\_freq}{2*(speed\_field+1)}$$

If the system clock is 250 MHz and the speed field is zero the SPI clock frequency is 125 MHz. The practical SPI clock will be lower as the I/O pads can not transmit or receive signals at such high speed. The lowest SPI clock frequency with a 250 MHz system clock is 30.5 KHz.

The hardware has an option to add hold time to the MOSI signal against the SPI clk. This is again done using the system clock. So a 250 MHz system clock will add hold times in units of 4 ns. Hold times of 0, 1, 4 and 7 system clock cycles can be used. (So at 250MHz an additional hold time of 0, 4, 16 and 28 ns can be achieved). The hold time is *additional* to the normal output timing as specified in the data sheet.

2.3.2 Interrupts

The SPI block has two interrupts: TX FIFO is empty, SPI is Idle.

TX FIFO is empty:
This interrupt will be asserted as soon as the last entry has been read from the transmit FIFO. At that time the interface will still be busy shifting out that data. This also implies that the receive FIFO will not yet contain the last received data. It is possible at that time to fill the TX FIFO again and read the receive FIFO entries which have been received. Note that there is no "receive FIFO full" interrupt as the number of entries received is always equal to the number of entries transmitted.

SPI is IDLE:
This interrupt will be asserted when the transmit FIFO is empty and the SPI block has finished all actions (including the CS-high time) By this time the receive FIFO will have all received data as well.

2.3.3 Long bit streams

The SPI module works in bursts of maximum 32 bits. Some SPI devices require data which is longer the 32 bits. To do this the user must make use of the two different data TX addresses: Tx data written to one address cause the CS to remain asserted. Tx data written to the other address cause the CS to be de-asserted at the end of the transmit cycle. So in order to exchange 96 bits you do the following:
Write the first two data words to one address, then write the third word to the other address.

2.3.4 SPI register details.

AUXSPI0/1_CNTL0 Register (0x7E21 5080,0x7E21 50C0)

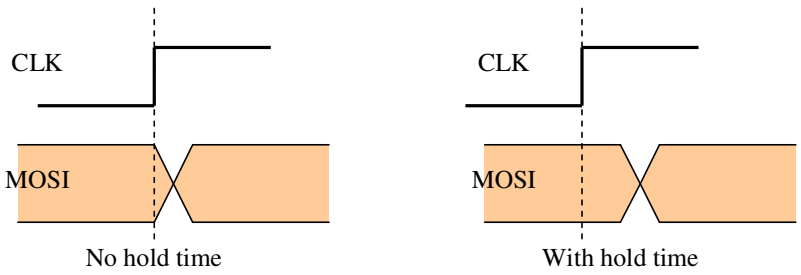
SYNOPSIS The AUXSPIx_CNTL0 register control many features of the SPI interfaces.

Bit(s)	Field Name	Description	Type	Reset
31:20	Speed	Sets the SPI clock speed. spi clk freq = system_clock_freq/2*(speed+1)	R/W	0
19:17	chip selects	The pattern output on the CS pins when active.	R/W	111
16	post-input mode	If set the SPI input works in post input mode. For details see text further down	R/W	0
15	Variable CS	If 1 the SPI takes the CS pattern and the data from the TX fifo If 0 the SPI takes the CS pattern from bits 17-19 of this register Set this bit only if also bit 14 (variable width) is set	R/W	0
14	Variable width	If 1 the SPI takes the shift length and the data from the TX fifo If 0 the SPI takes the shift length from bits 0-5 of this register	R/W	0
13:12	DOUT Hold time	Controls the extra DOUT hold time in system clock cycles. 00 : No extra hold time 01 : 1 system clock extra hold time 10 : 4 system clocks extra hold time 11 : 7 system clocks extra hold time	R/W	0
11	Enable	Enables the SPI interface. Whilst disabled the FIFOs can still be written to or read from This bit should be 1 during normal operation.	R/W	0
10	In rising	If 1 data is clocked in on the rising edge of the SPI clock If 0 data is clocked in on the falling edge of the SPI clock	R/W	0
9	Clear FIFOs	If 1 the receive and transmit FIFOs are held in reset (and thus flushed.) This bit should be 0 during normal operation.	R/W	0

8	Out rising	If 1 data is clocked out on the rising edge of the SPI clock If 0 data is clocked out on the falling edge of the SPI clock	R/W	0
7	Invert SPI CLK	If 1 the 'idle' clock line state is high. If 0 the 'idle' clock line state is low.	R/W	0
6	Shift out MS bit first	If 1 the data is shifted out starting with the MS bit. (bit 15 or bit 11) If 0 the data is shifted out starting with the LS bit. (bit 0)	R/W	0
5:0	Shift length	Specifies the number of bits to shift This field is ignored when using 'variable shift' mode	R/W	0

Invert SPI CLK
Changing this bit will immediately change the polarity of the SPI clock output. It is recommended not to do this when also the CS is active as the connected devices will see this as a clock change.

DOUT hold time
Because the interface runs of fast silicon the MOSI hold time against the clock will be very short. This can cause considerable problems on SPI slaves. To make it easier for the slave to see the data the hold time of the MOSI out against the SPI clock out is programmable.



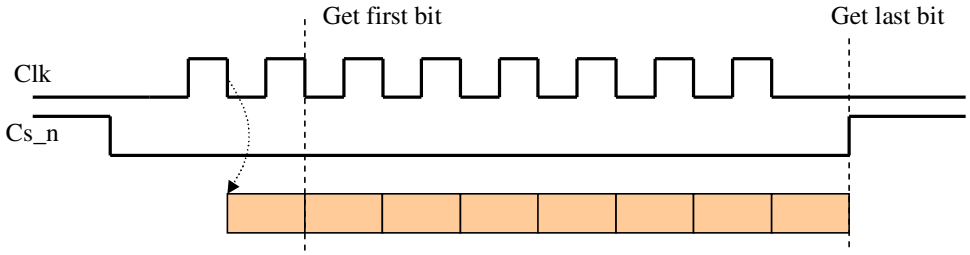
Variable width
In this mode the shift length is taken from the transmit FIFO. The transmit data bits 28:24 are used as shift length and the data bits 23:0 are the actual transmit data. If the option 'shift MS out first' is selected the first bit shifted out will be bit 23. The receive data will arrive as normal.

Variable CS
This mode is used together with the variable width mode. In this mode the CS pattern is taken from the transmit FIFO. The transmit data bits 31:29 are used as CS and the data bits 23:0 are the actual transmit data. This allows the CPU to write to different SPI devices without having to change the CS bits. However the data length is limited to 24 bits.

Post-input mode
Some rare SPI devices output data on the falling clock edge which then has to be picked up on the next falling clock edge. There are two problems with this:

1. The very first falling clock edge there is no valid data arriving.
2. After the last clock edge there is one more 'dangling' bit to pick up.

The post-input mode is specifically to deal with this sort of data. If the post-input mode bit is set, the data arriving at the first falling clock edge is ignored. Then after the last falling clock edge the CS remain asserted and after a full bit time the last data bit is picked up. The following figure shows this behaviour:



In this mode the CS will go high 1 full SPI clock cycle after the last clock edge. This guarantees a full SPI clock cycle time for the data to settle and arrive at the MISO input.

AUXSPI0/1_CNTL1 Register (0x7E21 5084,0x7E21 50C4)

SYNOPSIS The AUXSPIx_CNTL1 registers control more features of the SPI interfaces.

Bit(s)	Field Name	Description	Type	Reset
31:18	-	Reserved, write zero, read as don't care		
10:8	CS high time	Additional SPI clock cycles where the CS is high.	R/W	0
7	TX empty IRQ	If 1 the interrupt line is high when the transmit FIFO is empty	R/W	0
6	Done IRQ	If 1 the interrupt line is high when the interface is idle	R/W	0
5:2	-	Reserved, write zero, read as don't care		
1	Shift in MS bit first	If 1 the data is shifted in starting with the MS bit. (bit 15) If 0 the data is shifted in starting with the LS bit. (bit 0)	R/W	0
0	Keep input	If 1 the receiver shift register is NOT cleared. Thus new data is concatenated to old data. If 0 the receiver shift register is cleared before each transaction.	R/W	0

Keep input
Setting the 'Keep input' bit will make that the input shift register is not cleared between transactions. However the contents of the shift register is still written to the receive FIFO at the end of each transaction. E.g. if you receive two 8 bit values 0x81 followed by 0x46 the receive FIFO will contain: 0x0081 in the first entry and 0x8146 in the second entry. This mode may save CPU time concatenating bits (4 bits followed by 12 bits).

CS high time
The SPI CS will always be high for at least 1 SPI clock cycle. Some SPI devices need more time to process the data. This field will set a longer CS-high time. So the actual CS high time is (CS_high_time + 1) (In SPI clock cycles).

Interrupts
The SPI block has two interrupts: TX FIFO is empty, SPI is Idle.

TX FIFO is empty:
This interrupt will be asserted as soon as the last entry has been read from the transmit FIFO. At that time the interface will still be busy shifting out that data. This also implies that the receive FIFO will not yet contain the last received data.

It is possible at that time to fill the TX FIFO again and read the receive FIFO entries which have been received. There is a RX FIFO level field which tells exactly how many words are in the receive FIFO. In general at that time the receive FIFO should contain the number of Tx items minus one (the last one still being received). Note that there is no "receive FIFO full" interrupt or "receive FIFO overflow" flag as the number of entries received can never be more then the number of entries transmitted.

AUX is IDLE:
This interrupt will be asserted when the module has finished all activities, including waiting the minimum CS high time. This guarantees that any receive data will be available and `transparent' changes can be made to the configuration register (e.g. inverting the SPI clock polarity).

AUXSPI0/1_STAT Register (0x7E21 5088,0x7E21 50C8)

SYNOPSIS The AUXSPIx_STAT registers show the status of the SPI interfaces.

Bit(s)	Field Name	Description	Type	Reset
31:24	TX FIFO level	The number of data units in the transmit data FIFO	R/W	0
23:12	RX FIFO level	The number of data units in the receive data FIFO.	R/W	0
11:5	-	Reserved, write zero, read as don't care		
4	TX Full	If 1 the transmit FIFO is full If 0 the transmit FIFO can accept at least 1 data unit.	R/W	0
3	TX Empty	If 1 the transmit FIFO is empty If 0 the transmit FIFO holds at least 1 data unit.	R/W	0
2	RX Empty	If 1 the receiver FIFO is empty If 0 the receiver FIFO holds at least 1 data unit.	R/W	0
6	Busy	Indicates the module is busy transferring data.	R/W	0
5:0	Bit count	The number of bits still to be processed. Starts with 'shift-length' and counts down.	R/W	0

Busy
This status bit indicates if the module is busy. It will be clear when the TX FIFO is empty and the module has finished all activities, including waiting the minimum CS high time.

AUXSPI0/1_PEEK Register (0x7E21 508C,0x7E21 50CC)

SYNOPSIS The AUXSPIx_PEEK registers show received data of the SPI interfaces.

Bit(s)	Field Name	Description	Type	Reset
31:16	-	Reserved, write zero, read as don't care		
15:0	Data	Reads from this address will show the top entry from the receive FIFO, but the data is not taken from the FIFO. This provides a means of inspecting the data but not removing it from the FIFO.	RO	0

AUXSPI0/1_IO Register (0x7E21 50A0-0x7E21 50AC 0x7E21 50E0-0x7E21 50EC)

SYNOPSIS The AUXSPIx_IO registers are the primary data port of the SPI interfaces These four addresses all write to the same FIFO.

Writing to any of these addresses causes the SPI CS_n pins to be de-asserted at the end of the access

Bit(s)	Field Name	Description	Type	Reset
31:16	-	Reserved, write zero, read as don't care		
15:0	Data	Writes to this address range end up in the transmit FIFO. Data is lost when writing whilst the transmit FIFO is full. Reads from this address will take the top entry from the receive FIFO. Reading whilst the receive FIFO is will return the last data received.	R/W	0

AUXSPI0/1_TXHOLD Register

(0x7E21 50B0-0x7E21 50BC

0x7E21 50F0-0x7E21 50FC)

SYNOPSIS The AUXSPIx_TXHOLD registers are the extended CS port of the SPI interfaces These four addresses all write to the same FIFO.

Writing to these addresses causes the SPI CS_n pins to remain asserted at the end of the access

Bit(s)	Field Name	Description	Type	Reset
31:16	-	Reserved, write zero, read as don't care		
15:0	Data	Writes to this address range end up in the transmit FIFO. Data is lost when writing whilst the transmit FIFO is full. Reads from this address will take the top entry from the receive FIFO. Reading whilst the receive FIFO is will return the last data received.	R/W	0

3 BSC

3.1 Introduction

The Broadcom Serial Controller (BSC) controller is a master, fast-mode (400Kb/s) BSC controller. The Broadcom Serial Control bus is a proprietary bus compliant with the Philips® I2C bus/interface version 2.1 January 2000.

- I^2C single master only operation (supports clock stretching wait states)
- Both 7-bit and 10-bit addressing is supported.
  - Timing completely software controllable via registers

3.2 Register View

The BSC controller has eight memory-mapped registers. All accesses are assumed to be 32-bit. Note that the BSC2 master is used dedicated with the HDMI interface and should not be accessed by user programs.

There are three BSC masters inside BCM. The register addresses starts from

- BSC0: 0x7E20_5000
- BSC1: 0x7E80_4000
- BSC2 : 0x7E80_5000

The table below shows the address of I^2C interface where the address is an offset from one of the three base addreses listed above.

I2C Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	C	Control	32
0x4	S	Status	32
0x8	DLEN	Data Length	32
0xc	A	Slave Address	32
0x10	FIFO	Data FIFO	32
0x14	DIV	Clock Divider	32
0x18	DEL	Data Delay	32

0x1c	CLKT	Clock Stretch Timeout	32

C Register

Synopsis The control register is used to enable interrupts, clear the FIFO, define a read or write operation and start a transfer.
The READ field specifies the type of transfer.
The CLEAR field is used to clear the FIFO. Writing to this field is a one-shot operation which will always read back as zero. The CLEAR bit can set at the same time as the start transfer bit, and will result in the FIFO being cleared just prior to the start of transfer. Note that clearing the FIFO during a transfer will result in the transfer being aborted.
The ST field starts a new BSC transfer. This has a one shot action, and so the bit will always read back as 0 .
The INTD field enables interrupts at the end of a transfer the DONE condition. The interrupt remains active until the DONE condition is cleared by writing a 1 to the I2CS.DONE field. Writing a 0 to the INTD field disables interrupts on DONE.
The INTT field enables interrupts whenever the FIFO is or more empty and needs writing (i.e. during a write transfer) - the TXW condition. The interrupt remains active until the TXW condition is cleared by writing sufficient data to the FIFO to complete the transfer. Writing a 0 to the INTT field disables interrupts on TXW.
The INTR field enables interrupts whenever the FIFO is or more full and needs reading (i.e. during a read transfer) - the RXR condition. The interrupt remains active until the RXW condition is cleared by reading sufficient data from the RX FIFO. Writing a 0 to the INTR field disables interrupts on RXR.
The I2CEN field enables BSC operations. If this bit is 0 then transfers will not be performed. All register accesses are still permitted however.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		
15	I2CEN	I2C Enable 0 = BSC controller is disabled 1 = BSC controller is enabled	RW	0x0
14:11		***Reserved*** *- Write as 0, read as don't care*		
10	INTR	INTR Interrupt on RX 0 = Don t generate interrupts on RXR condition. 1 = Generate interrupt while RXR = 1.	RW	0x0
9	INTT	INTT Interrupt on TX 0 = Don t generate interrupts on TXW condition. 1 = Generate interrupt while TXW = 1.	RW	0x0

8	INTD	INTD Interrupt on DONE 0 = Don t generate interrupts on DONE condition. 1 = Generate interrupt while DONE = 1.	RW	0x0
7	ST	ST Start Transfer 0 = No action. 1 = Start a new transfer. One shot operation. Read back as 0.	RW	0x0
6		***Reserved*** *- Write as 0, read as don't care*		
5:4	CLEAR	CLEAR FIFO Clear 00 = No action. x1 = Clear FIFO. One shot operation. 1x = Clear FIFO. One shot operation. If CLEAR and ST are both set in the same operation, the FIFO is cleared before the new frame is started. Read back as 0. Note: 2 bits are used to maintain compatibility to previous version.	RW	0x0
3:1		***Reserved*** *- Write as 0, read as don't care*		
0	READ	READ Read Transfer 0 = Write Packet Transfer. 1 = Read Packet Transfer.	RW	0x0

S Register

Synopsis The status register is used to record activity status, errors and interrupt requests.
The TA field indicates the activity status of the BSC controller. This read-only field returns a 1 when the controller is in the middle of a transfer and a 0 when idle.
The DONE field is set when the transfer completes. The DONE condition can be used with I2CC.INTD to generate an interrupt on transfer completion. The DONE field is reset by writing a 1 , writing a 0 to the field has no effect.
The read-only TXW bit is set during a write transfer and the FIFO is less than full and needs writing. Writing sufficient data (i.e. enough data to either fill the FIFO more than full or complete the transfer) to the FIFO will clear the field. When the I2CC.INTT control bit is set, the TXW condition can be used to generate an interrupt to write more data to the FIFO to complete the current transfer. If the I2C controller runs out of data to send, it will wait for more data to be written into the FIFO.
The read-only RXR field is set during a read transfer and the FIFO is or more full and needs reading. Reading sufficient data to bring the depth below will clear the field. When I2CC.INTR control bit is set, the RXR condition can be used to generate an interrupt to read data from the FIFO before it becomes full. In the event that the FIFO does become full, all I2C operations will stall until data is removed from the FIFO.
The read-only TXD field is set when the FIFO has space for at least one byte of data. TXD is clear when the FIFO is full. The TXD field can be used to check that the FIFO can accept data before any is written. Any writes to a full TX FIFO will be ignored.
The read-only RXD field is set when the FIFO contains at least one byte of data. RXD is cleared when the FIFO becomes empty. The RXD field can be used to check that the FIFO contains data before reading. Reading from an empty FIFO will return invalid data.
The read-only TXE field is set when the FIFO is empty. No further data will be transmitted until more data is written to the FIFO.
The read-only RXF field is set when the FIFO is full. No more clocks will be generated until space is available in the FIFO to receive more data.
The ERR field is set when the slave fails to acknowledge either its address or a data byte written to it. The ERR field is reset by writing a 1 , writing a 0 to the field has no effect.
The CLKT field is set when the slave holds the SCL signal high for too long (clock stretching). The CLKT field is reset by writing a 1 , writing a 0 to the field has no effect.

Bit(s)	Field Name	Description	Type	Reset
31:10		***Reserved*** *- Write as 0, read as don't care*		
9	CLKT	CLKT Clock Stretch Timeout 0 = No errors detected. 1 = Slave has held the SCL signal low (clock stretching) for longer and that specified in the I2CCLKT register Cleared by writing 1 to the field.	RW	0x0
8	ERR	ERR ACK Error 0 = No errors detected. 1 = Slave has not acknowledged its address. Cleared by writing 1 to the field.	RW	0x0
7	RXF	RXF - FIFO Full 0 = FIFO is not full. 1 = FIFO is full. If a read is underway, no further serial data will be received until data is read from FIFO.	RO	0x0

6	TXE	TXE - FIFO Empty 0 = FIFO is not empty. 1 = FIFO is empty. If a write is underway, no further serial data can be transmitted until data is written to the FIFO.	RO	0x1
5	RXD	RXD - FIFO contains Data 0 = FIFO is empty. 1 = FIFO contains at least 1 byte. Cleared by reading sufficient data from FIFO.	RO	0x0
4	TXD	TXD - FIFO can accept Data 0 = FIFO is full. The FIFO cannot accept more data. 1 = FIFO has space for at least 1 byte.	RO	0x1
3	RXR	RXR - FIFO needs Reading (full) 0 = FIFO is less than full and a read is underway. 1 = FIFO is or more full and a read is underway. Cleared by reading sufficient data from the FIFO.	RO	0x0
2	TXW	TXW - FIFO needs Writing (full) 0 = FIFO is at least full and a write is underway (or sufficient data to send). 1 = FIFO is less then full and a write is underway. Cleared by writing sufficient data to the FIFO.	RO	0x0
1	DONE	DONE Transfer Done 0 = Transfer not completed. 1 = Transfer complete. Cleared by writing 1 to the field.	RW	0x0
0	TA	TA Transfer Active 0 = Transfer not active. 1 = Transfer active.	RO	0x0

DLEN Register

Synopsis The data length register defines the number of bytes of data to transmit or receive in the I2C transfer. Reading the register gives the number of bytes remaining in the current transfer.
The DLEN field specifies the number of bytes to be transmitted/received. Reading the DLEN field when a transfer is in progress (TA = 1) returns the number of bytes still to be transmitted or received. Reading the DLEN field when the transfer has just completed (DONE = 1) returns zero as there are no more bytes to transmit or receive. Finally, reading the DLEN field when TA = 0 and DONE = 0 returns the last value written. The DLEN field can be left over multiple transfers.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		

15:0	DLEN	Data Length. Writing to DLEN specifies the number of bytes to be transmitted/received. Reading from DLEN when TA = 1 or DONE = 1, returns the number of bytes still to be transmitted or received. Reading from DLEN when TA = 0 and DONE = 0, returns the last DLEN value written. DLEN can be left over multiple packets.	RW	0x0

A Register

Synopsis The slave address register specifies the slave address and cycle type. The address register can be left across multiple transfers
The ADDR field specifies the slave address of the I2C device.

Bit(s)	Field Name	Description	Type	Reset
31:7		***Reserved*** *- Write as 0, read as don't care*		
6:0	ADDR	Slave Address.	RW	0x0

FIFO Register

Synopsis The Data FIFO register is used to access the FIFO. Write cycles to this address place data in the 16-byte FIFO, ready to transmit on the BSC bus. Read cycles access data received from the bus.
Data writes to a full FIFO will be ignored and data reads from an empty FIFO will result in invalid data. The FIFO can be cleared using the I2CC.CLEAR field.
The DATA field specifies the data to be transmitted or received.

Bit(s)	Field Name	Description	Type	Reset
31:8		***Reserved*** *- Write as 0, read as don't care*		
7:0	DATA	Writes to the register write transmit data to the FIFO. Reads from register reads received data from the FIFO.	RW	0x0

DIV Register

Synopsis The clock divider register is used to define the clock speed of the BSC peripheral. The CDIV field specifies the core clock divider used by the BSC.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		
15:0	CDIV	Clock Divider SCL = core clock / CDIV Where core_clk is nominally 150 MHz. If CDIV is set to 0, the divisor is 32768. CDIV is always rounded down to an even number. The default value should result in a 100 kHz I2C clock frequency.	RW	0x5dc

DEL Register

Synopsis The data delay register provides fine control over the sampling/launch point of the data.
The REDL field specifies the number core clocks to wait after the rising edge before sampling the incoming data.
The FEDL field specifies the number core clocks to wait after the falling edge before outputting the next data bit.
Note: Care must be taken in choosing values for FEDL and REDL as it is possible to cause the BSC master to malfunction by setting values of CDIV/2 or greater. Therefore the delay values should always be set to less than CDIV/2.

Bit(s)	Field Name	Description	Type	Reset
31:16	FEDL	FEDL Falling Edge Delay Number of core clock cycles to wait after the falling edge of SCL before outputting next bit of data.	RW	0x30
15:0	REDL	REDL Rising Edge Delay Number of core clock cycles to wait after the rising edge of SCL before reading the next bit of data.	RW	0x30

CLKT Register

Synopsis The clock stretch timeout register provides a timeout on how long the master waits for the slave to stretch the clock before deciding that the slave has hung.
The TOUT field specifies the number I2C SCL clocks to wait after releasing SCL high and finding that the SCL is still low before deciding that the slave is not responding and moving the I2C machine forward. When a timeout occurs, the I2CS.CLKT bit is set.
Writing 0x0 to TOUT will result in the Clock Stretch Timeout being disabled.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		
15:0	TOUT	TOUT Clock Stretch Timeout Value Number of SCL clock cycles to wait after the rising edge of SCL before deciding that the slave is not responding.	RW	0x40

3.3 10 Bit Addressing

10 Bit addressing is an extension to the standard 7-bit addressing mode. This section describes in detail how to read/write using 10-bit addressing with this I2C controller.

10-bit addressing is compatible with, and can be combined with, 7 bit addressing. Using 10 bits for addressing exploits the reserved combination 1111 0xx for the first byte following a START (S) or REPEATED START (Sr) condition.

The 10 bit slave address is formed from the first two bytes following a S or Sr condition.

The first seven bits of the first byte are the combination 11110XX of which the last two bits (XX) are the two *most significant* bits of the 10-bit address. The eighth bit of the first byte is the R/W bit. If the R/W bit is '0' (write) then the following byte contains the remaining 8 bits of the 10-bit address. If the R/W bit is '1' then the next byte contains data transmitted from the slave to the master.

Writing

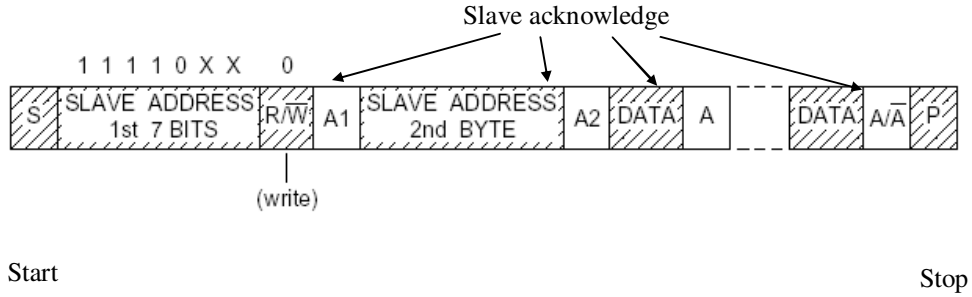


Figure 3-1 Write to a slave with 10 bit address

Figure 3-1 shows a write to a slave with a 10-bit address, to perform this using the controller one must do the following:

Assuming we are in the 'stop' state: (and the FIFO is empty)

1. Write the number of data bytes to written (plus one) to the I2CDLEN register.
2. Write 'XXXXXXXX' to the FIFO where 'XXXXXXXX' are the least 8 significant bits of the 10-bit slave address.
3. Write other data to be transmitted to the FIFO.
4. Write '11110XX' to Slave Address Register where 'XX' are the two most significant bits of the 10-bit address. Set I2CC.READ = 0 and I2CC.ST = 1, this will start a write transfer.

Reading

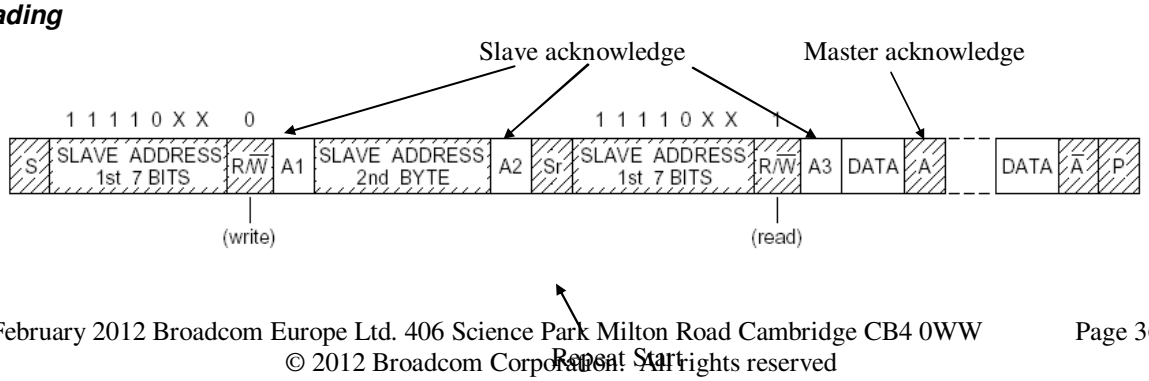


Figure 3-2 Read from slave with 10 bit address

Figure 3-2 shows how a read from a slave with a 10-bit address is performed. Following is the procedure for performing a read using the controller:

1. Write 1 to the I2CDLEN register.
2. Write 'XXXXXXXX' to the FIFO where 'XXXXXXXX' are the least 8 significant bits of the 10-bit slave address.
3. Write '11110XX' to the Slave Address Register where 'XX' are the two most significant bits of the 10-bit address. Set I2CC.READ = 0 and I2CC.ST = 1, this will start a write transfer.
4. Poll the I2CS.TA bit, waiting for the transfer has started.
5. Write the number of data bytes to read to the I2CDLEN register.
6. Set I2CC.READ = 1 and I2CC.ST = 1, this will send the repeat start bit, new slave address and R/W bit (which is '1') initiating the read.

4 DMA Controller

4.1 Overview

The majority of hardware pipelines and peripherals within the BCM2835 are bus masters, enabling them to efficiently satisfy their own data requirements. This reduces the requirements of the DMA controller to block-to-block memory transfers and supporting some of the simpler peripherals. In addition, the DMA controller provides a *read only* prefetch mode to allow data to be brought into the L2 cache in anticipation of its later use.

Beware that the DMA controller is direcly connected to the peripherals. Thus the DMA controller must be set-up to use the Physical (harware) addresses of the peripherals.

The BCM2835 DMA Controller provides a total of 16 DMA channels. Each channel operates independently from the others and is internally arbitrated onto one of the 3 system busses. This means that the amount of bandwidth that a DMA channel may consume can be controlled by the arbiter settings.

Each DMA channel operates by loading a *Control Block* (CB) data structure from memory into internal registers. The *Control Block* defines the required DMA operation. Each *Control Block* can point to a further *Control Block* to be loaded and executed once the operation described in the current *Control Block* has completed. In this way a linked list of *Control Blocks* can be constructed in order to execute a sequence of DMA operations without software intervention.

The DMA supports AXI read bursts to ensure efficient external SDRAM use. The DMA control block contains a burst parameter which indicates the required burst size of certain memory transfers. In general the DMA doesn't do write bursts, although wide writes will be done in 2 beat bursts if possible.

Memory-to-Peripheral transfers can be paced by a *Data Request* (DREQ) signal which is generated by the peripheral. The DREQ signal is level sensitive and controls the DMA by gating its AXI bus requests.

A peripheral can also provide a *Panic* signal alongside the DREQ to indicate that there is an imminent danger of FIFO underflow or overflow or similar critical situation. The *Panic* is used to select the AXI apriority level which is then passed out onto the AXI bus so that it can be used to influence arbitration in the rest of the system.

The allocation of peripherals to DMA channels is programmable.

The DMA can deal with byte aligned transfers and will minimise bus traffic by buffering and packing misaligned accesses.

Each DMA channel can be fully disabled via a top level power register to save power.

4.2 DMA Controller Registers

The DMA Controller is comprised of several identical DMA Channels depending upon the required configuration. Each individual DMA channel has an identical register map (although LITE channels have less functionality and hence less registers).

DMA Channel 0 is located at the address of 0x7E007000, Channel 1 at 0x7E007100, Channel 2 at 0x7E007200 and so on. Thus adjacent DMA Channels are offset by `0x100`.

DMA Channel 15 however, is physically removed from the other DMA Channels and so has a different address base of 0x7EE05000.

DMA Channel Offsets		
DMA Channels 0 – 14 Register Set Offsets from DMA0_BASE		
0x000		*DMA Channel 0 Register Set*
0x100		*DMA Channel 1 Register Set*
0x200		*DMA Channel 2 Register Set*
0x300		*DMA Channel 3 Register Set*
0x400		*DMA Channel 4 Register Set*
0x500		*DMA Channel 5 Register Set*
0x600		*DMA Channel 6 Register Set*
0x700		*DMA Channel 7 Register Set*
0x800		*DMA Channel 8 Register Set*
0x900		*DMA Channel 9 Register Set*
0xa00		*DMA Channel 10 Register Set*
0xb00		*DMA Channel 11 Register Set*
0xc00		*DMA Channel 12 Register Set*
0xd00		*DMA Channel 13 Register Set*
0xe00		*DMA Channel 14 Register Set*
DMA Channel 15 Register Set Offset from DMA15_BASE		
0x000		*DMA Channel 15 Register Set*

Table 4-1 – DMA Controller Register Address Map

4.2.1 DMA Channel Register Address Map

Each DMA channel has an identical register map, only the base address of each channel is different.

There is a global enable register at the top of the Address map that can disable each DMA for powersaving.

Only three registers in each channels register set are directly writeable (CS, CONBLK_AD *and DEBUG*). The other registers (TI, SOURCE_AD, DEST_AD, TXFR_LEN, STRIDE & NEXTCONBK), are automatically loaded from a *Control Block* data structure held in external memory.

4.2.1.1 Control Block Data Structure

Control Blocks (CB) are 8 words (256 bits) in length and must start at a 256-bit aligned address. The format of the CB data structure in memory, is shown below.

Each 32 bit word of the control block is automatically loaded into the corresponding 32 bit DMA control block register at the start of a DMA transfer. The descriptions of these registers also defines the corresponding bit locations in the CB data structure in memory.

32-bit Word Offset	Description	Associated Read-Only Register
0	Transfer Information	TI
1	Source Address	SOURCE_AD
2	Destination Address	DEST_AD
3	Transfer Length	TXFR_LEN
4	2D Mode Stride	STRIDE
5	Next Control Block Address	NEXTCONBK
6-7	*Reserved – set to zero.*	*N/A*

Table 4-2 – DMA Control Block Definition

The DMA is started by writing the address of a CB structure into the CONBLK_AD register and then setting the ACTIVE bit. The DMA will fetch the CB from the address set in the SCB_ADDR field of this reg and it will load it into the *read-only* registers described below. It will then begin a DMA transfer according to the information in the CB.

When it has completed the current DMA transfer (length => 0) the DMA will update the CONBLK_AD register with the contents of the NEXTCONBK register, fetch a new CB from that address, and start the whole procedure once again.

The DMA will stop (and clear the ACTIVE bit) when it has completed a DMA transfer and the NEXTCONBK register is set to `0x0000_0000`. It will load this value into the CONBLK_AD reg and then stop.

Most of the control block registers cannot be written to directly as they loaded automatically from memory. They can be read to provide status information, and to indicate the progress of the current DMA transfer. The value loaded into the NEXTCONBK register can be overwritten so that the linked list of Control Block data structures can be dynamically altered. However it is only safe to do this when the DMA is paused.

4.2.1.2 Register Map

DMA Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	0_CS	DMA Channel 0 Control and Status	32
0x4	0_CONBLK_AD	DMA Channel 0 Control Block Address	32
0x8	0_TI	DMA Channel 0 CB Word 0 (Transfer Information)	32
0xc	0_SOURCE_AD	DMA Channel 0 CB Word 1 (Source Address)	32
0x10	0_DEST_AD	DMA Channel 0 CB Word 2 (Destination Address)	32
0x14	0_TXFR_LEN	DMA Channel 0 CB Word 3 (Transfer Length)	32
0x18	0_STRIDE	DMA Channel 0 CB Word 4 (2D Stride)	32
0x1c	0_NEXTCONBK	DMA Channel 0 CB Word 5 (Next CB Address)	32
0x20	0_DEBUG	DMA Channel 0 Debug	32
0x100	1_CS	DMA Channel 1 Control and Status	32
0x104	1_CONBLK_AD	DMA Channel 1 Control Block Address	32
0x108	1_TI	DMA Channel 1 CB Word 0 (Transfer Information)	32
0x10c	1_SOURCE_AD	DMA Channel 1 CB Word 1 (Source Address)	32
0x110	1_DEST_AD	DMA Channel 1 CB Word 2 (Destination Address)	32
0x114	1_TXFR_LEN	DMA Channel 1 CB Word 3 (Transfer Length)	32

0x118	1_STRIDE	DMA Channel 1 CB Word 4 (2D Stride)	32
0x11c	1_NEXTCONBK	DMA Channel 1 CB Word 5 (Next CB Address)	32
0x120	1_DEBUG	DMA Channel 1 Debug	32
0x200	2_CS	DMA Channel 2 Control and Status	32
0x204	2_CONBLK_AD	DMA Channel 2 Control Block Address	32
0x208	2_TI	DMA Channel 2 CB Word 0 (Transfer Information)	32
0x20c	2_SOURCE_AD	DMA Channel 2 CB Word 1 (Source Address)	32
0x210	2_DEST_AD	DMA Channel 2 CB Word 2 (Destination Address)	32
0x214	2_TXFR_LEN	DMA Channel 2 CB Word 3 (Transfer Length)	32
0x218	2_STRIDE	DMA Channel 2 CB Word 4 (2D Stride)	32
0x21c	2_NEXTCONBK	DMA Channel 2 CB Word 5 (Next CB Address)	32
0x220	2_DEBUG	DMA Channel 2 Debug	32
0x300	3_CS	DMA Channel 3 Control and Status	32
0x304	3_CONBLK_AD	DMA Channel 3 Control Block Address	32
0x308	3_TI	DMA Channel 3 CB Word 0 (Transfer Information)	32
0x30c	3_SOURCE_AD	DMA Channel 3 CB Word 1 (Source Address)	32
0x310	3_DEST_AD	DMA Channel 3 CB Word 2 (Destination Address)	32
0x314	3_TXFR_LEN	DMA Channel 3 CB Word 3 (Transfer Length)	32
0x318	3_STRIDE	DMA Channel 3 CB Word 4 (2D Stride)	32
0x31c	3_NEXTCONBK	DMA Channel 3 CB Word 5 (Next CB Address)	32
0x320	3_DEBUG	DMA Channel 0 Debug	32

0x400	4_CS	DMA Channel 4 Control and Status	32
0x404	4_CONBLK_AD	DMA Channel 4 Control Block Address	32
0x408	4_TI	DMA Channel 4 CB Word 0 (Transfer Information)	32
0x40c	4_SOURCE_AD	DMA Channel 4 CB Word 1 (Source Address)	32
0x410	4_DEST_AD	DMA Channel 4 CB Word 2 (Destination Address)	32
0x414	4_TXFR_LEN	DMA Channel 4 CB Word 3 (Transfer Length)	32
0x418	4_STRIDE	DMA Channel 4 CB Word 4 (2D Stride)	32
0x41c	4_NEXTCONBK	DMA Channel 4 CB Word 5 (Next CB Address)	32
0x420	4_DEBUG	DMA Channel 0 Debug	32
0x500	5_CS	DMA Channel 5 Control and Status	32
0x504	5_CONBLK_AD	DMA Channel 5 Control Block Address	32
0x508	5_TI	DMA Channel 5 CB Word 0 (Transfer Information)	32
0x50c	5_SOURCE_AD	DMA Channel 5 CB Word 1 (Source Address)	32
0x510	5_DEST_AD	DMA Channel 5 CB Word 2 (Destination Address)	32
0x514	5_TXFR_LEN	DMA Channel 5 CB Word 3 (Transfer Length)	32
0x518	5_STRIDE	DMA Channel 5 CB Word 4 (2D Stride)	32
0x51c	5_NEXTCONBK	DMA Channel 5 CB Word 5 (Next CB Address)	32
0x520	5_DEBUG	DMA Channel 5 Debug	32
0x600	6_CS	DMA Channel 6 Control and Status	32
0x604	6_CONBLK_AD	DMA Channel 6 Control Block Address	32
0x608	6_TI	DMA Channel 6 CB Word 0 (Transfer Information)	32

0x60c	6_SOURCE_AD	DMA Channel 6 CB Word 1 (Source Address)	32
0x610	6_DEST_AD	DMA Channel 6 CB Word 2 (Destination Address)	32
0x614	6_TXFR_LEN	DMA Channel 6 CB Word 3 (Transfer Length)	32
0x618	6_STRIDE	DMA Channel 6 CB Word 4 (2D Stride)	32
0x61c	6_NEXTCONBK	DMA Channel 6 CB Word 5 (Next CB Address)	32
0x620	6_DEBUG	DMA Channel 6 Debug	32
0x700	7_CS	DMA Channel 7 Control and Status	32
0x704	7_CONBLK_AD	DMA Channel 7 Control Block Address	32
0x708	7_TI	DMA Channel 7 CB Word 0 (Transfer Information)	32
0x70c	7_SOURCE_AD	DMA Channel 7 CB Word 1 (Source Address)	32
0x710	7_DEST_AD	DMA Channel 7 CB Word 2 (Destination Address)	32
0x714	7_TXFR_LEN	DMA Channel 7 CB Word 3 (Transfer Length)	32
0x71c	7_NEXTCONBK	DMA Channel 7 CB Word 5 (Next CB Address)	32
0x720	7_DEBUG	DMA Channel 7 Debug	32
0x800	8_CS	DMA Channel 8 Control and Status	32
0x804	8_CONBLK_AD	DMA Channel 8 Control Block Address	32
0x808	8_TI	DMA Channel 8 CB Word 0 (Transfer Information)	32
0x80c	8_SOURCE_AD	DMA Channel 8 CB Word 1 (Source Address)	32
0x810	8_DEST_AD	DMA Channel 8 CB Word 2 (Destination Address)	32
0x814	8_TXFR_LEN	DMA Channel 8 CB Word 3 (Transfer Length)	32
0x81c	8_NEXTCONBK	DMA Channel 8 CB Word 5 (Next CB Address)	32

0x820	8_DEBUG	DMA Channel 8 Debug	32
0x900	9_CS	DMA Channel 9 Control and Status	32
0x904	9_CONBLK_AD	DMA Channel 9 Control Block Address	32
0x908	9_TI	DMA Channel 9 CB Word 0 (Transfer Information)	32
0x90c	9_SOURCE_AD	DMA Channel 9 CB Word 1 (Source Address)	32
0x910	9_DEST_AD	DMA Channel 9 CB Word 2 (Destination Address)	32
0x914	9_TXFR_LEN	DMA Channel 9 CB Word 3 (Transfer Length)	32
0x91c	9_NEXTCONBK	DMA Channel 9 CB Word 5 (Next CB Address)	32
0x920	9_DEBUG	DMA Channel 9 Debug	32
0xa00	10_CS	DMA Channel 10 Control and Status	32
0xa04	10_CONBLK_AD	DMA Channel 10 Control Block Address	32
0xa08	10_TI	DMA Channel 10 CB Word 0 (Transfer Information)	32
0xa0c	10_SOURCE_AD	DMA Channel 10 CB Word 1 (Source Address)	32
0xa10	10_DEST_AD	DMA Channel 10 CB Word 2 (Destination Address)	32
0xa14	10_TXFR_LEN	DMA Channel 10 CB Word 3 (Transfer Length)	32
0xa1c	10_NEXTCONBK	DMA Channel 10 CB Word 5 (Next CB Address)	32
0xa20	10_DEBUG	DMA Channel 10 Debug	32
0xb00	11_CS	DMA Channel 11 Control and Status	32
0xb04	11_CONBLK_AD	DMA Channel 11 Control Block Address	32
0xb08	11_TI	DMA Channel 11 CB Word 0 (Transfer Information)	32
0xb0c	11_SOURCE_AD	DMA Channel 11 CB Word 1 (Source Address)	32

0xb10	11_DEST_AD	DMA Channel 11 CB Word 2 (Destination Address)	32
0xb14	11_TXFR_LEN	DMA Channel 11 CB Word 3 (Transfer Length)	32
0xb1c	11_NEXTCONBK	DMA Channel 11 CB Word 5 (Next CB Address)	32
0xb20	11_DEBUG	DMA Channel 11 Debug	32
0xc00	12_CS	DMA Channel 12 Control and Status	32
0xc04	12_CONBLK_AD	DMA Channel 12 Control Block Address	32
0xc08	12_TI	DMA Channel 12 CB Word 0 (Transfer Information)	32
0xc0c	12_SOURCE_AD	DMA Channel 12 CB Word 1 (Source Address)	32
0xc10	12_DEST_AD	DMA Channel 12 CB Word 2 (Destination Address)	32
0xc14	12_TXFR_LEN	DMA Channel 12 CB Word 3 (Transfer Length)	32
0xc1c	12_NEXTCONBK	DMA Channel 12 CB Word 5 (Next CB Address)	32
0xc20	12_DEBUG	DMA Channel 12 Debug	32
0xd00	13_CS	DMA Channel 13 Control and Status	32
0xd04	13_CONBLK_AD	DMA Channel 13 Control Block Address	32
0xd08	13_TI	DMA Channel 13 CB Word 0 (Transfer Information)	32
0xd0c	13_SOURCE_AD	DMA Channel 13 CB Word 1 (Source Address)	32
0xd10	13_DEST_AD	DMA Channel 13 CB Word 2 (Destination Address)	32
0xd14	13_TXFR_LEN	DMA Channel 13 CB Word 3 (Transfer Length)	32
0xd1c	13_NEXTCONBK	DMA Channel 13 CB Word 5 (Next CB Address)	32
0xd20	13_DEBUG	DMA Channel 13 Debug	32
0xe00	14_CS	DMA Channel 14 Control and Status	32

0xe04	14_CONBLK_AD	DMA Channel 14 Control Block Address	32
0xe08	14_TI	DMA Channel 14 CB Word 0 (Transfer Information)	32
0xe0c	14_SOURCE_AD	DMA Channel 14 CB Word 1 (Source Address)	32
0xe10	14_DEST_AD	DMA Channel 14 CB Word 2 (Destination Address)	32
0xe14	14_TXFR_LEN	DMA Channel 14 CB Word 3 (Transfer Length)	32
0xe1c	14_NEXTCONBK	DMA Channel 14 CB Word 5 (Next CB Address)	32
0xe20	14_DEBUG	DMA Channel 14 Debug	32
0xfe0	INT_STATUS	Interrupt status of each DMA channel	32
0xff0	ENABLE	Global enable bits for each DMA channel	32

0_CS 1_CS 2_CS 3_CS 4_CS 5_CS 6_CS 7_CS 8_CS 9_CS 10_CS 11_CS 12_CS 13_CS 14_CS Register

Synopsis DMA Control And Status register contains the main control and status bits for this DMA channel.

Bit(s)	Field Name	Description	Type	Reset
31	RESET	DMA Channel Reset Writing a 1 to this bit will reset the DMA. The bit cannot be read, and will self clear.	W1SC	0x0
30	ABORT	Abort DMA Writing a 1 to this bit will abort the current DMA CB. The DMA will load the next CB and attempt to continue. The bit cannot be read, and will self clear.	W1SC	0x0
29	DISDEBUG	Disable debug pause signal When set to 1, the DMA will not stop when the debug pause signal is asserted.	RW	0x0

28	WAIT_FOR_OUTSTANDING_WRITES	Wait for outstanding writes When set to 1, the DMA will keep a tally of the AXI writes going out and the write responses coming in. At the very end of the current DMA transfer it will wait until the last outstanding write response has been received before indicating the transfer is complete. Whilst waiting it will load the next CB address (but will not fetch the CB), clear the active flag (if the next CB address = zero), and it will defer setting the END flag or the INT flag until the last outstanding write response has been received. In this mode, the DMA will pause if it has more than 13 outstanding writes at any one time.	RW	0x0
27:24		***Reserved*** *- Write as 0, read as don't care*		
23:20	PANIC_PRIORITY	AXI Panic Priority Level Sets the priority of panicking AXI bus transactions. This value is used when the panic bit of the selected peripheral channel is 1. Zero is the lowest priority.	RW	0x0
19:16	PRIORITY	AXI Priority Level Sets the priority of normal AXI bus transactions. This value is used when the panic bit of the selected peripheral channel is zero. Zero is the lowest priority.	RW	0x0
15:9		***Reserved*** *- Write as 0, read as don't care*		
8	ERROR	DMA Error Indicates if the DMA has detected an error. The error flags are available in the debug register, and have to be cleared by writing to that register. 1 = DMA channel has an error flag set. 0 = DMA channel is ok.	RO	0x0
7		***Reserved*** *- Write as 0, read as don't care*		

6	WAITING_FOR_OUTSTANDING_WRITES	DMA is Waiting for the Last Write to be Received Indicates if the DMA is currently waiting for any outstanding writes to be received, and is not transferring data. 1 = DMA channel is waiting.	RO	0x0
5	DREQ_STOPS_DMA	DMA Paused by DREQ State Indicates if the DMA is currently paused and not transferring data due to the DREQ being inactive.. 1 = DMA channel is paused. 0 = DMA channel is running.	RO	0x0
4	PAUSED	DMA Paused State Indicates if the DMA is currently paused and not transferring data. This will occur if: the active bit has been cleared, if the DMA is currently executing wait cycles or if the debug_pause signal has been set by the debug block, or the number of outstanding writes has exceeded the max count. 1 = DMA channel is paused. 0 = DMA channel is running.	RO	0x0
3	DREQ	DREQ State Indicates the state of the selected DREQ (Data Request) signal, ie. the DREQ selected by the PERMAP field of the transfer info. 1 = Requesting data. This will only be valid once the DMA has started and the PERMAP field has been loaded from the CB. It will remain valid, indicating the selected DREQ signal, until a new CB is loaded. If PERMAP is set to zero (un-paced transfer) then this bit will read back as 1. 0 = No data request.	RO	0x0
2	INT	Interrupt Status This is set when the transfer for the CB ends and INTEN is set to 1. Once set it must be manually cleared down, even if the next CB has INTEN = 0. Write 1 to clear.	W1C	0x0

1	END	DMA End Flag Set when the transfer described by the current control block is complete. Write 1 to clear.	W1C	0x0
0	ACTIVE	Activate the DMA This bit enables the DMA. The DMA will start if this bit is set and the CB_ADDR is non zero. The DMA transfer can be paused and resumed by clearing, then setting it again. This bit is automatically cleared at the end of the complete DMA transfer, ie. after a NEXTCONBK = 0x0000_0000 has been loaded.	RW	0x0

0_CONBLK_AD 1_CONBLK_AD 2_CONBLK_AD 3_CONBLK_AD 4_CONBLK_AD 5_CONBLK_AD 6_CONBLK_AD 7_CONBLK_AD 8_CONBLK_AD 9_CONBLK_AD 10_CONBLK_AD 11_CONBLK_AD 12_CONBLK_AD 13_CONBLK_AD 14_CONBLK_AD Register

Synopsis DMA Control Block Address register.

Bit(s)	Field Name	Description	Type	Reset
31:0	SCB_ADDR	Control Block Address This tells the DMA where to find a Control Block stored in memory. When the ACTIVE bit is set and this address is non zero, the DMA will begin its transfer by loading the contents of the addressed CB into the relevant DMA channel registers. At the end of the transfer this register will be updated with the ADDR field of the NEXTCONBK control block register. If this field is zero, the DMA will stop. Reading this register will return the address of the currently active CB (in the linked list of CB s). The address must be 256 bit aligned, so the bottom 5 bits of the address must be zero.	RW	0x0

0_TI 1_TI 2_TI 3_TI 4_TI 5_TI 6_TI Register

Synopsis DMA Transfer Information.

Bit(s)	Field Name	Description	Type	Reset

31:27		***Reserved*** *- Write as 0, read as don't care*		
26	NO_WIDE_BURSTS	Don t Do wide writes as a 2 beat burst This prevents the DMA from issuing wide writes as 2 beat AXI bursts. This is an inefficient access mode, so the default is to use the bursts.	RW	0x0
25:21	WAITS	Add Wait Cycles This slows down the DMA throughput by setting the number of dummy cycles burnt after each DMA read or write operation is completed. A value of 0 means that no wait cycles are to be added.	RW	0x0
20:16	PERMAP	Peripheral Mapping Indicates the peripheral number (1-31) whose ready signal shall be used to control the rate of the transfers, and whose panic signals will be output on the DMA AXI bus. Set to 0 for a continuous un-paced transfer.	RW	0x0
15:12	BURST_LENGTH	Burst Transfer Length Indicates the burst length of the DMA transfers. The DMA will attempt to transfer data as bursts of this number of words. A value of zero will produce a single transfer. Bursts are only produced for specific conditions, see main text.	RW	0x0
11	SRC_IGNORE	Ignore Reads 1 = Do not perform source reads. In addition, destination writes will zero all the write strobes. This is used for fast cache fill operations. 0 = Perform source reads..	RW	0x0
10	SRC_DREQ	Control Source Reads with DREQ 1 = The DREQ selected by PER_MAP will gate the source reads. 0 = DREQ has no effect.	RW	0x0
9	SRC_WIDTH	Source Transfer Width 1 = Use 128-bit source read width. 0 = Use 32-bit source read width.	RW	0x0
8	SRC_INC	Source Address Increment 1 = Source address increments after each read. The address will increment by 4, if S_WIDTH=0 else by 32. 0 = Source address does not change.	RW	0x0

7	DEST_IGNORE	Ignore Writes 1 = Do not perform destination writes. 0 = Write data to destination.	RW	0x0
6	DEST_DREQ	Control Destination Writes with DREQ 1 = The DREQ selected by PERMAP will gate the destination writes. 0 = DREQ has no effect.	RW	0x0
5	DEST_WIDTH	Destination Transfer Width 1 = Use 128-bit destination write width. 0 = Use 32-bit destination write width.	RW	0x0
4	DEST_INC	Destination Address Increment 1 = Destination address increments after each write The address will increment by 4, if DEST_WIDTH=0 else by 32. 0 = Destination address does not change.	RW	0x0
3	WAIT_RESP	Wait for a Write Response When set this makes the DMA wait until it receives the AXI write response for each write. This ensures that multiple writes cannot get stacked in the AXI bus pipeline. 1= Wait for the write response to be received before proceeding. 0 = Don t wait; continue as soon as the write data is sent.	RW	0x0
2		***Reserved*** *- Write as 0, read as don't care*		
1	TDMODE	2D Mode 1 = 2D mode interpret the TXFR_LEN register as YLENGTH number of transfers each of XLENGTH, and add the strides to the address after each transfer. 0 = Linear mode interpret the TXFR register as a single transfer of total length {YLENGTH ,XLENGTH}.	RW	0x0
0	INTEN	Interrupt Enable 1 = Generate an interrupt when the transfer described by the current Control Block completes. 0 = Do not generate an interrupt.	RW	0x0

0_SOURCE_AD 1_SOURCE_AD 2_SOURCE_AD 3_SOURCE_AD 4_SOURCE_AD 5_SOURCE_AD 6_SOURCE_AD 7_SOURCE_AD 8_SOURCE_AD 9_SOURCE_AD 10_SOURCE_AD 11_SOURCE_AD 12_SOURCE_AD 13_SOURCE_AD 14_SOURCE_AD Register

Synopsis DMA Source Address

Bit(s)	Field Name	Description	Type	Reset
31:0	S_ADDR	DMA Source Address Source address for the DMA operation. Updated by the DMA engine as the transfer progresses.	RW	0x0

0_DEST_AD 1_DEST_AD 2_DEST_AD 3_DEST_AD 4_DEST_AD 5_DEST_AD 6_DEST_AD 7_DEST_AD 8_DEST_AD 9_DEST_AD 10_DEST_AD 11_DEST_AD 12_DEST_AD 13_DEST_AD 14_DEST_AD Register

Synopsis DMA Destination Address

Bit(s)	Field Name	Description	Type	Reset
31:0	D_ADDR	DMA Destination Address Destination address for the DMA operation. Updated by the DMA engine as the transfer progresses.	RW	0x0

0_TXFR_LEN 1_TXFR_LEN 2_TXFR_LEN 3_TXFR_LEN 4_TXFR_LEN 5_TXFR_LEN 6_TXFR_LEN Register

Synopsis DMA Transfer Length. This specifies the amount of data to be transferred in bytes.
In normal (non 2D) mode this specifies the amount of bytes to be transferred.
In 2D mode it is interpreted as an X and a Y length, and the DMA will perform Y transfers, each of length X bytes and add the strides onto the addresses after each X leg of the transfer.
The length register is updated by the DMA engine as the transfer progresses, so it will indicate the data left to transfer.

Bit(s)	Field Name	Description	Type	Reset
31:30		***Reserved*** *- Write as 0, read as don't care*		
29:16	YLENGTH	When in 2D mode, This is the Y transfer length, indicating how many xlength transfers are performed. When in normal linear mode this becomes the top bits of the XLENGTH	RW	0x0
15:0	XLENGTH	Transfer Length in bytes.	RW	0x0

0_STRIDE 1_STRIDE 2_STRIDE 3_STRIDE 4_STRIDE 5_STRIDE 6_STRIDE Register

Synopsis DMA 2D Stride

Bit(s)	Field Name	Description	Type	Reset
31:16	D_STRIDE	Destination Stride (2D Mode) Signed (2 s complement) byte increment to apply to the destination address at the end of each row in 2D mode.	RW	0x0
15:0	S_STRIDE	Source Stride (2D Mode) Signed (2 s complement) byte increment to apply to the source address at the end of each row in 2D mode.	RW	0x0

0_NEXTCONBK 1_NEXTCONBK 2_NEXTCONBK 3_NEXTCONBK 4_NEXTCONBK 5_NEXTCONBK 6_NEXTCONBK 7_NEXTCONBK 8_NEXTCONBK 9_NEXTCONBK 10_NEXTCONBK 11_NEXTCONBK 12_NEXTCONBK 13_NEXTCONBK 14_NEXTCONBK Register

Synopsis DMA Next Control Block Address
The value loaded into this register can be overwritten so that the linked list of Control Block data structures can be altered. However it is only safe to do this when the DMA is paused. The address must be 256 bit aligned and so the bottom 5 bits cannot be set and will read back as zero.

Bit(s)	Field Name	Description	Type	Reset
31:0	ADDR	Address of next CB for chained DMA operations.	RW	0x0

0_DEBUG 1_DEBUG 2_DEBUG 3_DEBUG 4_DEBUG 5_DEBUG 6_DEBUG Register

Synopsis DMA Debug register.

Bit(s)	Field Name	Description	Type	Reset
31:29		***Reserved*** *- Write as 0, read as don't care*		

28	LITE	DMA Lite Set if the DMA is a reduced performance LITE engine.	RO	0x0
27:25	VERSION	DMA Version DMA version number, indicating control bit filed changes.	RO	0x2
24:16	DMA_STATE	DMA State Machine State Returns the value of the DMA engines state machine for this channel.	RO	0x0
15:8	DMA_ID	DMA ID Returns the DMA AXI ID of this DMA channel.	RO	0x0
7:4	OUTSTANDING_WRITES	DMA Outstanding Writes Counter Returns the number of write responses that have not yet been received. This count is reset at the start of each new DMA transfer or with a DMA reset.	RO	0x0
3		***Reserved*** *- Write as 0, read as don't care*		
2	READ_ERROR	Slave Read Response Error Set if the read operation returned an error value on the read response bus. It can be cleared by writing a 1,	RW	0x0
1	FIFO_ERROR	Fifo Error Set if the optional read Fifo records an error condition. It can be cleared by writing a 1,	RW	0x0
0	READ_LAST_NOT_SET_ERROR	Read Last Not Set Error If the AXI read last signal was not set when expected, then this error bit will be set. It can be cleared by writing a 1.	RW	0x0

7_TI 8_TI 9_TI 10_TI 11_TI 12_TI 13_TI 14_TI Register

Synopsis DMA Transfer Information.

Bit(s)	Field Name	Description	Type	Reset
31:26		***Reserved*** *- Write as 0, read as don't care*		

25:21	WAITS	Add Wait Cycles This slows down the DMA throughput by setting the number of dummy cycles burnt after each DMA read or write operation is completed. A value of 0 means that no wait cycles are to be added.	RW	0x0
20:16	PERMAP	Peripheral Mapping Indicates the peripheral number (1-31) whose ready signal shall be used to control the rate of the transfers, and whose panic signals will be output on the DMA AXI bus. Set to 0 for a continuous un-paced transfer.	RW	0x0
15:12	BURST_LENGTH	Burst Transfer Length Indicates the burst length of the DMA transfers. The DMA will attempt to transfer data as bursts of this number of words. A value of zero will produce a single transfer. Bursts are only produced for specific conditions, see main text.	RW	0x0
11	SRC_IGNORE		RW	0x0
10	SRC_DREQ	Control Source Reads with DREQ 1 = The DREQ selected by PER_MAP will gate the source reads. 0 = DREQ has no effect.	RW	0x0
9	SRC_WIDTH	Source Transfer Width 1 = Use 128-bit source read width. 0 = Use 32-bit source read width.	RW	0x0
8	SRC_INC	Source Address Increment 1 = Source address increments after each read. The address will increment by 4, if S_WIDTH=0 else by 32. 0 = Source address does not change.	RW	0x0
7	DEST_IGNORE		RW	0x0
6	DEST_DREQ	Control Destination Writes with DREQ 1 = The DREQ selected by PERMAP will gate the destination writes. 0 = DREQ has no effect.	RW	0x0
5	DEST_WIDTH	Destination Transfer Width 1 = Use 128-bit destination write width. 0 = Use 32-bit destination write width.	RW	0x0

4	DEST_INC	Destination Address Increment 1 = Destination address increments after each write The address will increment by 4, if DEST_WIDTH=0 else by 32. 0 = Destination address does not change.	RW	0x0
3	WAIT_RESP	Wait for a Write Response When set this makes the DMA wait until it receives the AXI write response for each write. This ensures that multiple writes cannot get stacked in the AXI bus pipeline. 1= Wait for the write response to be received before proceeding. 0 = Don t wait; continue as soon as the write data is sent.	RW	0x0
2:1		***Reserved*** *- Write as 0, read as don't care*		
0	INTEN	Interrupt Enable 1 = Generate an interrupt when the transfer described by the current Control Block completes. 0 = Do not generate an interrupt.	RW	0x0

7_TXFR_LEN 8_TXFR_LEN 9_TXFR_LEN 10_TXFR_LEN 11_TXFR_LEN 12_TXFR_LEN 13_TXFR_LEN 14_TXFR_LEN Register

Synopsis DMA Transfer Length

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		
15:0	XLENGTH	Transfer Length Length of transfer, in bytes. Updated by the DMA engine as the transfer progresses.	RW	0x0

7_DEBUG 8_DEBUG 9_DEBUG 10_DEBUG 11_DEBUG 12_DEBUG 13_DEBUG 14_DEBUG Register

Synopsis DMA Lite Debug register.

Bit(s)	Field Name	Description	Type	Reset

31:29		***Reserved*** *- Write as 0, read as don't care*		
28	LITE	DMA Lite Set if the DMA is a reduced performance LITE engine.	RO	0x1
27:25	VERSION	DMA Version DMA version number, indicating control bit filed changes.	RO	0x2
24:16	DMA_STATE	DMA State Machine State Returns the value of the DMA engines state machine for this channel.	RO	0x0
15:8	DMA_ID	DMA ID Returns the DMA AXI ID of this DMA channel.	RO	0x0
7:4	OUTSTANDING_WRITES	DMA Outstanding Writes Counter Returns the number of write responses that have not yet been received. This count is reset at the start of each new DMA transfer or with a DMA reset.	RO	0x0
3		***Reserved*** *- Write as 0, read as don't care*		
2	READ_ERROR	Slave Read Response Error Set if the read operation returned an error value on the read response bus. It can be cleared by writing a 1,	RW	0x0
1	FIFO_ERROR	Fifo Error Set if the optional read Fifo records an error condition. It can be cleared by writing a 1,	RW	0x0
0	READ_LAST_NOT_SET_ERROR	Read Last Not Set Error If the AXI read last signal was not set when expected, then this error bit will be set. It can be cleared by writing a 1.	RW	0x0

INT_STATUS Register

Synopsis Interrupt status of each DMA engine

Bit(s)	Field Name	Description	Type	Reset

31:16		***Reserved*** *- Write as 0, read as don't care*		
15	INT15	Interrupt status of DMA engine 15	RW	0x0
14	INT14	Interrupt status of DMA engine 14	RW	0x0
13	INT13	Interrupt status of DMA engine 13	RW	0x0
12	INT12	Interrupt status of DMA engine 12	RW	0x0
11	INT11	Interrupt status of DMA engine 11	RW	0x0
10	INT10	Interrupt status of DMA engine 10	RW	0x0
9	INT9	Interrupt status of DMA engine 9	RW	0x0
8	INT8	Interrupt status of DMA engine 8	RW	0x0
7	INT7	Interrupt status of DMA engine 7	RW	0x0
6	INT6	Interrupt status of DMA engine 6	RW	0x0
5	INT5	Interrupt status of DMA engine 5	RW	0x0
4	INT4	Interrupt status of DMA engine 4	RW	0x0
3	INT3	Interrupt status of DMA engine 3	RW	0x0
2	INT2	Interrupt status of DMA engine 2	RW	0x0
1	INT1	Interrupt status of DMA engine 1	RW	0x0
0	INT0	Interrupt status of DMA engine 0	RW	0x0

ENABLE Register

Synopsis Global enable bits for each channel

Bit(s)	Field Name	Description	Type	Reset
31:15		***Reserved*** *- Write as 0, read as don't care*		

14	EN14	enable dma engine 14	RW	0x1
13	EN13	enable dma engine 13	RW	0x1
12	EN12	enable dma engine 12	RW	0x1
11	EN11	enable dma engine 11	RW	0x1
10	EN10	enable dma engine 10	RW	0x1
9	EN9	enable dma engine 9	RW	0x1
8	EN8	enable dma engine 8	RW	0x1
7	EN7	enable dma engine 7	RW	0x1
6	EN6	enable dma engine 6	RW	0x1
5	EN5	enable dma engine 5	RW	0x1
4	EN4	enable dma engine 4	RW	0x1
3	EN3	enable dma engine 3	RW	0x1
2	EN2	enable dma engine 2	RW	0x1
1	EN1	enable dma engine 1	RW	0x1
0	EN0	enable dma engine 0	RW	0x1

4.2.1.3 Peripheral DREQ Signals

A DREQ (Data Request) mechanism is used to pace the data flow between the DMA and a peripheral.

Each peripheral is allocated a permanent DREQ signal. Each DMA channel can select which of the DREQ signals should be used to pace the transfer by controlling the DMA reads, DMA writes or both. Note that DREQ 0 is permanently enabled and can be used if no DREQ is required.

When a DREQ signal is being used to pace the DMA reads, the DMA will wait until it has sampled DREQ high before launching a single or burst read operation. It will then wait for all the read data to be returned before re-checking the DREQ and starting the next read. Thus once a peripheral receives the read request it should remove its DREQ as soon as possible to prevent the DMA from re-sampling the same DREQ assertion.

DREQ's are not required when reading from AXI peripherals. In this case, the DMA will request data from the peripheral and the peripheral will only send the data when it is available. The DMA will not request data that is does not have room for, so no pacing of the data flow is required.

DREQ's are required when reading from APB peripherals as the AXI-to-APB bridge will not wait for an APB peripheral to be ready and will just perfom the APB read regardless. Thus an APB peripheral needs to make sure that it has all of its read data ready before it drives its DREQ high.

When writing to peripherals, a DREQ is always required to pace the data. However, due to the pipelined nature of the AXI bus system, several writes may be in flight before the peripheral receives any data and withdraws its DREQ signal. Thus the peripheral must ensure that it has sufficient room in its input FIFO to accommodate the maximum amount of data that it might receive. If the peripheral is unable to do this, the DMA WAIT_RESP mechanism can be used to ensure that only one write is in flight at any one time, however this is less efficient transfer mechanism.

The mapping of peripherals to DREQ's is as follows:

DREQ	Peripheral
0	DREQ = 1 This is always on so use this channel if no DREQ is required.
1	DSI
2	PCM TX
3	PCM RX
4	SMI
5	PWM
6	SPI TX
7	SPI RX

8	BSC/SPI Slave TX
9	BSC/SPI Slave RX
10	unused
11	*e.MMC*
12	*UART TX*
13	SD HOST
14	UART RX.
15	DSI
16	SLIMBUS MCTX.
17	HDMI
18	SLIMBUS MCRX
19	SLIMBUS DC0
20	SLIMBUS DC1
21	SLIMBUS DC2
22	SLIMBUS DC3
23	SLIMBUS DC4
24	Scaler FIFO 0 & SMI *
25	Scaler FIFO 1 & SMI *
26	Scaler FIFO 2 & SMI *
27	SLIMBUS DC5
28	SLIMBUS DC6
29	SLIMBUS DC7
30	SLIMBUS DC8
31	SLIMBUS DC9

* The SMI element of the Scaler FIFO 0 & SMI DREQs can be disabled by setting the SMI_DISABLE bit in the DMA_DREQ_CONTROL register in the system arbiter control block.

4.3 AXI Bursts

The DMA supports bursts under specific conditions. Up to 16 beat bursts can be accommodated.

Peripheral (32 bit wide) read bursts are supported. The DMA will generate the burst if there is sufficient room in its read buffer to accommodate all the data from the burst. This limits the burst size to a maximum of 8 beats.

Read bursts in destination ignore mode (DEST_IGNORE) are supported as there is no need for the DMA to deal with the data. This allows wide bursts of up to 16 beats to be used for efficient L2 cache fills.

DMA channel 0 and 15 are fitted with an external 128 bit 8 word read FIFO. This enables efficient memory to memory transfers to be performed. This FIFO allows the DMA to accommodate a wide read burst up to the size of the FIFO. In practice this will allow a 128 bit wide read burst of 9 as the first word back will be immediately read into the DMA engine (or a 32 bit peripheral read burst of 16 – 8 in the input buffer and 8 in the fifo). On any DMA channel, if a read burst is selected that is too large, the AXI read bus will be stalled until the DMA has written out the data. This may lead to inefficient system operation, and possibly AXI lock up if it causes a circular dependancy.

In general write bursts are not supported. However to increase the efficiency of L2 cache fills, src_ignore (SRC_IGNORE) transfers can be specified with a write burst. In this case the DMA will issue a write burst address sequence followed by the appropriate number of zero data, zero strobe write bus cycles, which will cause the cache to pre-fetch the data. To improve the efficiency of the 128 bit wide bus architecture, and to make use of the DMAs internal 256 bit registers, the DMA will generate 128 bit wide writes as 2 beat bursts wherever possible, although this behaviour can be disabled.

4.4 Error Handling

If the DMA detects a Read Response error it will record the fact in the READ_ERROR flag in the debug register. This will remain set until it is cleared by writing a 1 to it. The DMA will clear its active flag and generate an interrupt. Any outstanding read data transactions (remainder of a burst) will be honoured. This allows the operator to either restart the DMA by clearing the error bit and setting the active bit, or to abort the DMA transfer by clearing the NEXTCONBK register and restarting the DMA with the ABORT bit set.

The DMA will also record any errors from an external read FIFO. These will be latched in the FIFO_ERROR bit in the debug register until they are cleared by writing a ‘1’ to the bit. (note that only DMA0 and 15 have an external read fifo)

If the DMA detects that a read occurred without the AXI rlast set as expected then it will set the READ_LAST_NOT_SET_ERROR bit in the debug register. This can be cleared by writing a ‘1’ to it.

The error bits are logically OR’d together and presented as a general ERROR bit in the CS register.

4.5 DMA LITE Engines

Several of the DMA engines are of the LITE design. This is a reduced specification engine designed to save space. The engine behaves in the same way as a normal DMA engine except for the following differences.

1. The internal data structure is 128 bits instead of 256 bits. This means that if you do a 128 bit wide read burst of more than 1 beat, the DMA input register will be full and the read bus will be stalled. The normal DMA engine can accept a read burst of 2 without stalling. If you do a narrow 32 bit read burst from the peripherals then the lite engine can cope with a burst of 4 as opposed to a burst of 8 for the normal engine. Note that stalling the read bus will potentially reduce the overall system performance, and may possible cause a system lockup if you end up with a conflict where the DMA cannot free the read bus as the read stall has prevented it writing out its data due to some circular system relationship.
2. The Lite engine does not support 2D transfers. The TDMODE, S_STRIDE, D_STRIDE and YLENGTH registers will all be removed. Setting these registers will have no effect.
3. The DMA length register is now 16 bits, limiting the maximum transferrable length to 65536 bytes.
4. Source ignore (SRC_IGNORE) and destination ignore (DEST_IGNORE) modes are removed.

The Lite engine will have about half the bandwidth of a normal DMA engine, and are intended for low bandwith peripheral servicing.

5 External Mass Media Controller

o Introduction

The External Mass Media Controller (EMMC) is an embedded MultiMedia™ and SD™ card interface provided by Arasan™. It is compliant to the following standards:

- SD™ Host Controller Standard Specification Version 3.0 Draft 1.0
- SDIO™ card specification version 3.0
- SD™ Memory Card Specification Draft version 3.0
- SD™ Memory Card Security Specification version 1.01
- MMC™ Specification version 3.31,4.2 and 4.4

For convenience in the following text card is used as a placeholder for SD™, embedded MultiMedia and SDIO™ cards.

For detailed information about the EMMC internals please refer to the Arasan™ document SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf but make sure to read the following chapter which lists the changes made to Arasan™'s IP.

Because the EMMC module shares pins with other functionality it must be selected in the GPIO interface. Please refer to the GPIO section for further details.

The interface to the card uses its own clock clk_emmc which is provided by the clock manager module. The frequency of this clock should be selected between 50 MHz and 100 MHz. Having a separate clock allows high performance access to the card even if the VideoCore runs at a reduced clock frequency. The EMMC module contains its own internal clock divider to generate the card's clock from clk_emmc.

Additionally can the sampling clock for the response and data from the card be delayed in up to 40 steps with a configurable delay between 200ps to 1100ps per step typically. The delay is intended to cancel the internal delay inside the card (up to 14ns) when reading. The delay per step will vary with temperature and supply voltage. Therefore it is better to use a bigger delay than necessary as there is no restriction for the maximum delay.

The EMMC module handles the handshaking process on the command and data lines and all CRC processing automatically.

Command execution is commenced by writing the command plus the appropriate flags to the CMDTM register after loading any required argument into the ARG1 register. The EMMC module calculates the CRC checksum, transfers the command to the card, receives the response and checks its CRC. Once the command has executed or timed-out bit 0 of register INTERRUPT will be set. Please note that the INTERRUPT register is not self clearing, so the software has first to reset it by writing 1 before using it to detect if a command has finished.

The software is responsible for checking the status bits of the card's response in order to verify successful processing by the card.

In order to transfer data from/to the card register DATA is accessed after configuring the host and sending the according commands to the card using CMDTM. Because the EMMC module doesn't interpret the commands sent to the card it is important to configure it identical to the card setup using the CONTROL0 register. Especial care should be taken to make sure that the width of the data bus is configured identical for host and card. The card is synchronized to the data flow by switching off its clock appropriately. A handshake signal dma_req is available for paced data transfers. Bit 1 of the INTERRUPT register can used to determine whether a data transfer has finished. Please note that the INTERRUPT register is not self clearing, so the software has first to reset it by writing 1 before using it to detect if a data transfer has finished.

The EMMC module restricts the maximum block size to the size of the internal data FIFO which is 1k bytes. In order to get maximum performance for data transfers it is necessary to use multiple block data transfers. In this case the EMMC module uses two FIFOs in ping-pong mode, i.e. one is used to transfer data to/from the card while the other is simultaneously accessed by DMA via the AXI bus. If the EMMC module is configured for single block transfers only one FIFO is used, so no DMA access is possible while data is transferred to/from the card and vice versa resulting in long dead times.

○ Registers

Contrary to Arasan™'s documentation the EMMC module registers can only be accessed as 32 bit registers, i.e. the two LSBs of the address are always zero.

The EMMC register base address is 0x7E300000

EMMC Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	ARG2	ACMD23 Argument	32
0x4	BLKSIZECNT	Block Size and Count	32
0x8	ARG1	Argument	32
0xc	CMDTM	Command and Transfer Mode	32
0x10	RESP0	Response bits 31 : 0	32
0x14	RESP1	Response bits 63 : 32	32

0x18	RESP2	Response bits 95 : 64	32
0x1c	RESP3	Response bits 127 : 96	32
0x20	DATA	Data	32
0x24	STATUS	Status	32
0x28	CONTROL0	Host Configuration bits	32
0x2c	CONTROL1	Host Configuration bits	32
0x30	INTERRUPT	Interrupt Flags	32
0x34	IRPT_MASK	Interrupt Flag Enable	32
0x38	IRPT_EN	Interrupt Generation Enable	32
0x3c	CONTROL2	Host Configuration bits	32
0x50	FORCE_IRPT	Force Interrupt Event	32
0x70	BOOT_TIMEOUT	Timeout in boot mode	32
0x74	DBG_SEL	Debug Bus Configuration	32
0x80	EXRDFIFO_CFG	Extension FIFO Configuration	32
0x84	EXRDFIFO_EN	Extension FIFO Enable	32
0x88	TUNE_STEP	Delay per card clock tuning step	32
0x8c	TUNE_STEPS_STD	Card clock tuning steps for SDR	32
0x90	TUNE_STEPS_DDR	Card clock tuning steps for DDR	32
0xf0	SPI_INT_SPT	SPI Interrupt Support	32
0xfc	SLOTISR_VER	Slot Interrupt Status and Version	32

ARG2 Register

Synopsis This register contains the argument for the SD card specific command ACMD23 (SET_WR_BLK_ERASE_COUNT). ARG2 must be set before the ACMD23 command is issued using the CMDTM register.

Bit(s)	Field Name	Description	Type	Reset
31:0	ARGUMENT	Argument to be issued with ACMD23	RW	0x0

BLKSIZECNT Register

Synopsis This register must not be accessed or modified while any data transfer between card and host is ongoing.
It contains the number and size in bytes for data blocks to be transferred. Please note that the EMMC module restricts the maximum block size to the size of the internal data FIFO which is 1k bytes.
BLKCNT is used to tell the host how many blocks of data are to be transferred. Once the data transfer has started and the TM_BLKCNT_EN bit in the CMDTM register is set the EMMC module automatically decreases the BNTCNT value as the data blocks are transferred and stops the transfer once BLKCNT reaches 0.

Bit(s)	Field Name	Description	Type	Reset
31:16	BLKCNT	Number of blocks to be transferred	RW	0x0
15:10		***Reserved*** *- Write as 0, read as don't care*		
9:0	BLKSIZE	Block size in bytes	RW	0x0

ARG1 Register

Synopsis This register contains the arguments for all commands except for the SD card specific command ACMD23 which uses ARG2. ARG1 must be set before the command is issued using the CMDTM register.

Bit(s)	Field Name	Description	Type	Reset
31:0	ARGUMENT	Argument to be issued with command	RW	0x0

CMDTM Register

Synopsis This register is used to issue commands to the card. Besides the command it also contains flags informing the EMMC module what card response and type of data transfer to expect. Incorrect flags will result in strange behaviour.
For data transfers two modes are supported: either transferring a single block of data or several blocks of the same size. The SD card uses two different sets of commands to differentiate between them but the host needs to be additionally configured using TM_MULTI_BLOCK. It is important that this bit is set correct for the command sent to the card, i.e. 1 for CMD18 and CMD25 and 0 for CMD17 and CMD24. Multiple block transfer gives a better performance.
The BLKSIZECNT register is used to configure the size and number of blocks to be transferred. If bit TM_BLKCNT_EN of this register is set the transfer stops automatically after the number of data blocks configured in the BLKSIZECNT register has been transferred.
The TM_AUTO_CMD_EN bits can be used to make the host to send automatically a command to the card telling it that the data transfer has finished once the BLKCNT bits in the BLKSIZECNT register are 0.

Bit(s)	Field Name	Description	Type	Reset
31:30		***Reserved*** *- Write as 0, read as don't care*		
29:24	CMD_INDEX	Index of the command to be issued to the card	RW	0x0
23:22	CMD_TYPE	Type of command to be issued to the card: 00 = normal 01 = suspend (the current data transfer) 10 = resume (the last data transfer) 11 = abort (the current data transfer)	RW	0x0
21	CMD_ISDATA	Command involves data transfer: 0 = no data transfer command 1 = data transfer command	RW	0x0
20	CMD_IXCHK_EN	Check that response has same index as command: 0 = disabled 1 = enabled	RW	0x0
19	CMD_CRCCHK_EN	Check the responses CRC: 0 = disabled 1 = enabled	RW	0x0
18		***Reserved*** *- Write as 0, read as don't care*		
17:16	CMD_RSPNS_TYPE	Type of expected response from card: 00 = no response 01 = 136 bits response 10 = 48 bits response 11 = 48 bits response using busy	RW	0x0

15:6		***Reserved*** *- Write as 0, read as don't care*		
5	TM_MULTI_BLOCK	Type of data transfer 0 = single block 1 = multiple block	RW	0x0
4	TM_DAT_DIR	Direction of data transfer: 0 = from host to card 1 = from card to host	RW	0x0
3:2	TM_AUTO_CMD_EN	Select the command to be send after completion of a data transfer: 00 = no command 01 = command CMD12 10 = command CMD23 11 = reserved	RW	0x0
1	TM_BLKCNT_EN	Enable the block counter for multiple block transfers: 0 = disabled 1 = enabled	RW	0x0
0		***Reserved*** *- Write as 0, read as don't care*		

RESP0 Register

Synopsis This register contains the status bits of the SD card s response. In case of commands CMD2 and CMD10 it contains CID[31:0] and in case of command CMD9 it contains CSD[31:0].
Note: this register is only valid once the last command has completed and no new command was issued.

Bit(s)	Field Name	Description	Type	Reset
31:0	RESPONSE	Bits 31:0 of the card s response	RW	0x0

RESP1 Register

Synopsis In case of commands CMD2 and CMD10 this register contains CID[63:32] and in case of command CMD9 it contains CSD[63:32].
Note: this register is only valid once the last command has completed and no new command was issued.

Bit(s)	Field Name	Description	Type	Reset
31:0	RESPONSE	Bits 63:32 of the card s response	RW	0x0

RESP2 Register

Synopsis In case of commands CMD2 and CMD10 this register contains CID[95:64] and in case of command CMD9 it contains CSD[95:64].
Note: this register is only valid once the last command has completed and no new command was issued.

Bit(s)	Field Name	Description	Type	Reset
31:0	RESPONSE	Bits 95:64 of the card s response	RW	0x0

RESP3 Register

Synopsis In case of commands CMD2 and CMD10 this register contains CID[127:96] and in case of command CMD9 it contains CSD[127:96].
Note: this register is only valid once the last command has completed and no new command was issued.

Bit(s)	Field Name	Description	Type	Reset
31:0	RESPONSE	Bits 127:96 of the card s response	RW	0x0

DATA Register

Synopsis This register is used to transfer data to/from the card.
Bit 1 of the INTERRUPT register can be used to check if data is available. For paced DMA transfers the high active signal dma_req can be used.

Bit(s)	Field Name	Description	Type	Reset
31:0	DATA	Data to/from the card	RW	0x0

STATUS Register

Synopsis This register contains information intended for debugging. Its values change automatically according to the hardware. As it involves resynchronisation between different clock domains it changes only after some latency and it is easy sample the values too early.
Therefore it is not recommended to use this register for polling. Instead use the INTERRUPT register which implements a handshake mechanism which makes it impossible to miss a change when polling.

Bit(s)	Field Name	Description	Type	Reset
31:29		***Reserved*** *- Write as 0, read as don't care*		
28:25	DAT_LEVEL1	Value of data lines DAT7 to DAT4	RW	0xf
24	CMD_LEVEL	Value of command line CMD	RW	0x1
23:20	DAT_LEVEL0	Value of data lines DAT3 to DAT0	RW	0xf
19:10		***Reserved*** *- Write as 0, read as don't care*		
9	READ_TRANSFER	New data can be read from EMMC: 0 = no 1 = yes	RW	0x0
8	WRITE_TRANSFER	New data can be written to EMMC: 0 = no 1 = yes	RW	0x0
7:3		***Reserved*** *- Write as 0, read as don't care*		
2	DAT_ACTIVE	At least one data line is active: 0 = no 1 = yes	RW	0x0
1	DAT_INHIBIT	Data lines still used by previous data transfer: 0 = no 1 = yes	RW	0x0
0	CMD_INHIBIT	Command line still used by previous command: 0 = no 1 = yes	RW	0x0

CONTROL0 Register

Synopsis This register is used to configure the EMMC module.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

Bit(s)	Field Name	Description	Type	Reset
31:23		***Reserved*** *- Write as 0, read as don't care*		
22	ALT_BOOT_EN	Enable alternate boot mode access: 0 = disabled 1 = enabled	RW	0x0
21	BOOT_EN	Boot mode access: 0 = stop boot mode access 1 = start boot mode access	RW	0x0
20	SPI_MODE	SPI mode enable: 0 = normal mode 1 = SPI mode	RW	0x0
19	GAP_IEN	Enable SDIO interrupt at block gap (only valid if the HCTL_DWIDTH bit is set): 0 = disabled 1 = enabled	RW	0x0
18	READWAIT_EN	Use DAT2 read-wait protocol for SDIO cards supporting this: 0 = disabled 1 = enabled	RW	0x0
17	GAP_RESTART	Restart a transaction which was stopped using the GAP_STOP bit: 0 = ignore 1 = restart	RW	0x0
16	GAP_STOP	Stop the current transaction at the next block gap: 0 = ignore 1 = stop	RW	0x0
15:6		***Reserved*** *- Write as 0, read as don't care*		
5	HCTL_8BIT	Use 8 data lines: 0 = disabled 1 = enabled	RW	0x0

4:3		***Reserved*** *- Write as 0, read as don't care*		
2	HCTL_HS_EN	Select high speed mode (i.e. DAT and CMD lines change on the rising CLK edge): 0 = disabled 1 = enabled	RW	0x0
1	HCTL_DWIDTH	Use 4 data lines: 0 = disabled 1 = enabled	RW	0x0
0		***Reserved*** *- Write as 0, read as don't care*		

CONTROL1 Register

Synopsis This register is used to configure the EMMC module.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.
CLK_STABLE seems contrary to its name only to indicate that there was a rising edge on the clk_emmc input but not that the frequency of this clock is actually stable.

Bit(s)	Field Name	Description	Type	Reset
31:27		***Reserved*** *- Write as 0, read as don't care*		
26	SRST_DATA	Reset the data handling circuit: 0 = disabled 1 = enabled	RW	0x0
25	SRST_CMD	Reset the command handling circuit: 0 = disabled 1 = enabled	RW	0x0
24	SRST_HC	Reset the complete host circuit: 0 = disabled 1 = enabled	RW	0x0
23:20		***Reserved*** *- Write as 0, read as don't care*		
19:16	DATA_TOUNIT	Data timeout unit exponent: 1111 = disabled x = TMCLK * 2^(x+13)	RW	0x0
15:8	CLK_FREQ8	SD clock base divider LSBs	RW	0x0

7:6	CLK_FREQ_MS2	SD clock base divider MSBs	RW	0x0
5	CLK_GENSEL	Mode of clock generation: 0 = divided 1 = programmable	RW	0x0
4:3		***Reserved*** *- Write as 0, read as don't care*		
2	CLK_EN	SD clock enable: 0 = disabled 1 = enabled	RW	0x0
1	CLK_STABLE	SD clock stable: 0 = no 1 = yes	RO	0x0
0	CLK_INTLEN	Clock enable for internal EMMC clocks for power saving: 0 = disabled 1 = enabled	RW	0x0

INTERRUPT Register

Synopsis This register holds the interrupt flags. Each flag can be disabled using the according bit in the IRPT_MASK register.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.
ERR is a generic flag and is set if any of the enabled error flags is set.

Bit(s)	Field Name	Description	Type	Reset
31:25		***Reserved*** *- Write as 0, read as don't care*		
24	ACMD_ERR	Auto command error: 0 = no error 1 = error	RW	0x0
23		***Reserved*** *- Write as 0, read as don't care*		
22	DEND_ERR	End bit on data line not 1: 0 = no error 1 = error	RW	0x0

21	DCRC_ERR	Data CRC error: 0 = no error 1 = error	RW	0x0
20	DTO_ERR	Timeout on data line: 0 = no error 1 = error	RW	0x0
19	CBAD_ERR	Incorrect command index in response: 0 = no error 1 = error	RW	0x0
18	CEND_ERR	End bit on command line not 1: 0 = no error 1 = error	RW	0x0
17	CCRC_ERR	Command CRC error: 0 = no error 1 = error	RW	0x0
16	CTO_ERR	Timeout on command line: 0 = no error 1 = error	RW	0x0
15	ERR	An error has occured: 0 = no error 1 = error	RO	0x0
14	ENDBOOT	Boot operation has terminated: 0 = no 1 = yes	RW	0x0
13	BOOTACK	Boot acknowledge has been received: 0 = no 1 = yes	RW	0x0
12	RETUNE	Clock retune request was made: 0 = no 1 = yes	RO	0x0
11:9		***Reserved*** *- Write as 0, read as don't care*		
8	CARD	Card made interrupt request: 0 = no 1 = yes	RO	0x0
7:6		***Reserved*** *- Write as 0, read as don't care*		

5	READ_RDY	DATA register contains data to be read: 0 = no 1 = yes	RW	0x0
4	WRITE_RDY	Data can be written to DATA register: 0 = no 1 = yes	RW	0x0
3		***Reserved*** *- Write as 0, read as don't care*		
2	BLOCK_GAP	Data transfer has stopped at block gap: 0 = no 1 = yes	RW	0x0
1	DATA_DONE	Data transfer has finished: 0 = no 1 = yes	RW	0x0
0	CMD_DONE	Command has finished: 0 = no 1 = yes	RW	0x0

IRPT_MASK Register

Synopsis This register is used to mask the interrupt flags in the INTERRUPT register. For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

Bit(s)	Field Name	Description	Type	Reset
31:25		***Reserved*** *- Write as 0, read as don't care*		
24	ACMD_ERR	Set flag if auto command error: 0 = no 1 = yes	RW	0x0
23		***Reserved*** *- Write as 0, read as don't care*		
22	DEND_ERR	Set flag if end bit on data line not 1: 0 = no 1 = yes	RW	0x0

21	DCRC_ERR	Set flag if data CRC error: 0 = no 1 = yes	RW	0x0
20	DTO_ERR	Set flag if timeout on data line: 0 = no 1 = yes	RW	0x0
19	CBAD_ERR	Set flag if incorrect command index in response: 0 = no 1 = yes	RW	0x0
18	CEND_ERR	Set flag if end bit on command line not 1: 0 = no 1 = yes	RW	0x0
17	CCRC_ERR	Set flag if command CRC error: 0 = no 1 = yes	RW	0x0
16	CTO_ERR	Set flag if timeout on command line: 0 = no 1 = yes	RW	0x0
15		***Reserved*** *- Write as 0, read as don't care*		
14	ENDBOOT	Set flag if boot operation has terminated: 0 = no 1 = yes	RW	0x0
13	BOOTACK	Set flag if boot acknowledge has been received: 0 = no 1 = yes	RW	0x0
12	RETUNE	Set flag if clock retune request was made: 0 = no 1 = yes	RW	0x0
11:9		***Reserved*** *- Write as 0, read as don't care*		
8	CARD	Set flag if card made interrupt request: 0 = no 1 = yes	RW	0x0
7:6		***Reserved*** *- Write as 0, read as don't care*		
5	READ_RDY	Set flag if DATA register contains data to be read: 0 = no 1 = yes	RW	0x0

4	WRITE_RDY	Set flag if data can be written to DATA register: 0 = no 1 = yes	RW	0x0
3		***Reserved*** *- Write as 0, read as don't care*		
2	BLOCK_GAP	Set flag if data transfer has stopped at block gap: 0 = no 1 = yes	RW	0x0
1	DATA_DONE	Set flag if data transfer has finished: 0 = no 1 = yes	RW	0x0
0	CMD_DONE	Set flag if command has finished: 0 = no 1 = yes	RW	0x0

IRPT_EN Register

Synopsis This register is used to enable the different interrupts in the INTERRUPT register to generate an interrupt on the int_to_arm output.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

Bit(s)	Field Name	Description	Type	Reset
31:25		***Reserved*** *- Write as 0, read as don't care*		
24	ACMD_ERR	Create interrupt if auto command error: 0 = no 1 = yes	RW	0x0
23		***Reserved*** *- Write as 0, read as don't care*		
22	DEND_ERR	Create interrupt if end bit on data line not 1: 0 = no 1 = yes	RW	0x0
21	DCRC_ERR	Create interrupt if data CRC error: 0 = no 1 = yes	RW	0x0

20	DTO_ERR	Create interrupt if timeout on data line: 0 = no 1 = yes	RW	0x0
19	CBAD_ERR	Create interrupt if incorrect command index in response: 0 = no 1 = yes	RW	0x0
18	CEND_ERR	Create interrupt if end bit on command line not 1: 0 = no 1 = yes	RW	0x0
17	CCRC_ERR	Create interrupt if command CRC error: 0 = no 1 = yes	RW	0x0
16	CTO_ERR	Create interrupt if timeout on command line: 0 = no 1 = yes	RW	0x0
15		***Reserved*** *- Write as 0, read as don't care*		
14	ENDBOOT	Create interrupt if boot operation has terminated: 0 = no 1 = yes	RW	0x0
13	BOOTACK	Create interrupt if boot acknowledge has been received: 0 = no 1 = yes	RW	0x0
12	RETUNE	Create interrupt if clock retune request was made: 0 = no 1 = yes	RW	0x0
11:9		***Reserved*** *- Write as 0, read as don't care*		
8	CARD	Create interrupt if card made interrupt request: 0 = no 1 = yes	RW	0x0
7:6		***Reserved*** *- Write as 0, read as don't care*		
5	READ_RDY	Create interrupt if DATA register contains data to be read: 0 = no 1 = yes	RW	0x0

4	WRITE_RDY	Create interrupt if data can be written to DATA register: 0 = no 1 = yes	RW	0x0
3		***Reserved*** *- Write as 0, read as don't care*		
2	BLOCK_GAP	Create interrupt if data transfer has stopped at block gap: 0 = no 1 = yes	RW	0x0
1	DATA_DONE	Create interrupt if data transfer has finished: 0 = no 1 = yes	RW	0x0
0	CMD_DONE	Create interrupt if command has finished: 0 = no 1 = yes	RW	0x0

CONTROL2 Register

Synopsis This register is used to enable the different interrupts in the INTERRUPT register to generate an interrupt on the int_to_arm output.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

Bit(s)	Field Name	Description	Type	Reset
31:24		***Reserved*** *- Write as 0, read as don't care*		
23	TUNED	Tuned clock is used for sampling data: 0 = no 1 = yes	RW	0x0
22	TUNEON	Start tuning the SD clock: 0 = not tuned or tuning complete 1 = tuning	RW	0x0
21:19		***Reserved*** *- Write as 0, read as don't care*		

18:16	UHSMODE	Select the speed mode of the SD card: 000 = SDR12 001 = SDR25 010 = SDR50 011 = SDR104 100 = DDR50 other = reserved	RW	0x0
15:8		***Reserved*** *- Write as 0, read as don't care*		
7	NOTC12_ERR	Error occurred during auto command CMD12 execution: 0 = no error 1 = error	RO	0x0
6:5		***Reserved*** *- Write as 0, read as don't care*		
4	ACBAD_ERR	Command index error occurred during auto command execution: 0 = no error 1 = error	RO	0x0
3	ACEND_ERR	End bit is not 1 during auto command execution: 0 = no error 1 = error	RO	0x0
2	ACCRC_ERR	Command CRC error occurred during auto command execution: 0 = no error 1 = error	RO	0x0
1	ACTO_ERR	Timeout occurred during auto command execution: 0 = no error 1 = error	RO	0x0
0	ACNOX_ERR	Auto command not executed due to an error: 0 = no 1 = yes	RO	0x0

FORCE_IRPT Register

Synopsis This register is used to fake the different interrupt events for debugging. For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

Bit(s)	Field Name	Description	Type	Reset
31:25		***Reserved*** *- Write as 0, read as don't care*		
24	ACMD_ERR	Create auto command error: 0 = no 1 = yes	RW	0x0
23		***Reserved*** *- Write as 0, read as don't care*		
22	DEND_ERR	Create end bit on data line not 1: 0 = no 1 = yes	RW	0x0
21	DCRC_ERR	Create data CRC error: 0 = no 1 = yes	RW	0x0
20	DTO_ERR	Create timeout on data line: 0 = no 1 = yes	RW	0x0
19	CBAD_ERR	Create incorrect command index in response: 0 = no 1 = yes	RW	0x0
18	CEND_ERR	Create end bit on command line not 1: 0 = no 1 = yes	RW	0x0
17	CCRC_ERR	Create command CRC error: 0 = no 1 = yes	RW	0x0
16	CTO_ERR	Create timeout on command line: 0 = no 1 = yes	RW	0x0
15		***Reserved*** *- Write as 0, read as don't care*		
14	ENDBOOT	Create boot operation has terminated: 0 = no 1 = yes	RW	0x0
13	BOOTACK	Create boot acknowledge has been received: 0 = no 1 = yes	RW	0x0

12	RETUNE	Create clock retune request was made: 0 = no 1 = yes	RW	0x0
11:9		***Reserved*** *- Write as 0, read as don't care*		
8	CARD	Create card made interrupt request: 0 = no 1 = yes	RW	0x0
7:6		***Reserved*** *- Write as 0, read as don't care*		
5	READ_RDY	Create DATA register contains data to be read: 0 = no 1 = yes	RW	0x0
4	WRITE_RDY	Create data can be written to DATA register: 0 = no 1 = yes	RW	0x0
3		***Reserved*** *- Write as 0, read as don't care*		
2	BLOCK_GAP	Create interrupt if data transfer has stopped at block gap: 0 = no 1 = yes	RW	0x0
1	DATA_DONE	Create data transfer has finished: 0 = no 1 = yes	RW	0x0
0	CMD_DONE	Create command has finished: 0 = no 1 = yes	RW	0x0

BOOT_TIMEOUT Register

Synopsis This register configures after how many card clock cycles a timeout for e.MMC cards in boot mode is flagged

Bit(s)	Field Name	Description	Type	Reset
31:0	TIMEOUT	Number of card clock cycles after which a timeout during boot mode is flagged	RW	0x0

DBG_SEL Register

Synopsis This register selects which submodules are accessed by the debug bus.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bits marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

Bit(s)	Field Name	Description	Type	Reset
31:1		***Reserved*** *- Write as 0, read as don't care*		
0	SELECT	Submodules accessed by debug bus: 0 = receiver and fifo_ctrl 1 = others	RW	0x0

EXRDFIFO_CFG Register

Synopsis This register allows fine tuning the dma_req generation for paced DMA transfers when reading from the card. If the extension data FIFO contains less than RD_THRSH 32 bits words dma_req becomes inactive until the card has filled the extension data FIFO above threshold. This compensates the DMA latency.
When writing data to the card the extension data FIFO feeds into the EMMC module s FIFO and no fine tuning is required Therefore the RD_THRSH value is in this case ignored.

Bit(s)	Field Name	Description	Type	Reset
31:3		***Reserved*** *- Write as 0, read as don't care*		
2:0	RD_THRSH	Read threshold in 32 bits words	RW	0x0

EXRDFIFO_EN Register

Synopsis This register enables the extension data register. It should be enabled for paced DMA transfers and be bypassed for burst DMA transfers.

Bit(s)	Field Name	Description	Type	Reset
31:1		***Reserved*** *- Write as 0, read as don't care*		

0	ENABLE	Enable the extension FIFO: 0 = bypass 1 = enabled	RW	0x0

TUNE_STEP Register

Synopsis This register is used to delay the card clock when sampling the returning data and command response from the card.
DELAY determines by how much the sampling clock is delayed per step.

Bit(s)	Field Name	Description	Type	Reset
31:3		***Reserved*** *- Write as 0, read as don't care*		
2:0	DELAY	Sampling clock delay per step: 000 = 200ps typically 001 = 400ps typically 010 = 400ps typically 011 = 600ps typically 100 = 700ps typically 101 = 900ps typically 110 = 900ps typically 111 = 1100ps typically	RW	0x0

TUNE_STEPS_STD Register

Synopsis This register is used to delay the card clock when sampling the returning data and command response from the card. It determines by how many steps the sampling clock is delayed in SDR mode.

Bit(s)	Field Name	Description	Type	Reset
31:6		***Reserved*** *- Write as 0, read as don't care*		
5:0	STEPS	Number of steps (0 to 40)	RW	0x0

TUNE_STEPS_DDR Register

Synopsis This register is used to delay the card clock when sampling the returning data and command response from the card. It determines by how many steps the sampling clock is delayed in DDR mode.

Bit(s)	Field Name	Description	Type	Reset
31:6		***Reserved*** *- Write as 0, read as don't care*		
5:0	STEPS	Number of steps (0 to 40)	RW	0x0

SPI_INT_SPT Register

Synopsis This register controls whether assertion of interrupts in SPI mode is possible independent of the card select line.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bit marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

Bit(s)	Field Name	Description	Type	Reset
31:8		***Reserved*** *- Write as 0, read as don't care*		
7:0	SELECT	Interrupt independent of card select line: 0 = no 1 = yes	RW	0x0

SLOTISR_VER Register

Synopsis This register contains the version information and slot interrupt status.
For the exact details please refer to the Arasan documentation SD3.0_Host_AHB_eMMC4.4_Usersguide_ver5.9_jan11_10.pdf. Bit marked as reserved in this document but not by the Arasan documentation refer to functionality which has been disabled due to the changes listed in the previous chapter.

Bit(s)	Field Name	Description	Type	Reset
31:24	VENDOR	Vendor Version Number	RW	0x0
23:16	SDVERSION	Host Controller specification version	RW	0x0

15:8		***Reserved*** *- Write as 0, read as don't care*		
7:0	SLOT_STATUS	Logical OR of interrupt and wakeup signal for each slot	RW	0x0

6 General Purpose I/O (GPIO)

There are 54 general-purpose I/O (GPIO) lines split into two banks. All GPIO pins have at least two alternative functions within BCM. The alternate functions are usually peripheral IO and a single peripheral may appear in each bank to allow flexibility on the choice of IO voltage. Details of alternative functions are given in section 6.2. Alternative Function Assignments.

The block diagram for an individual GPIO pin is given below :

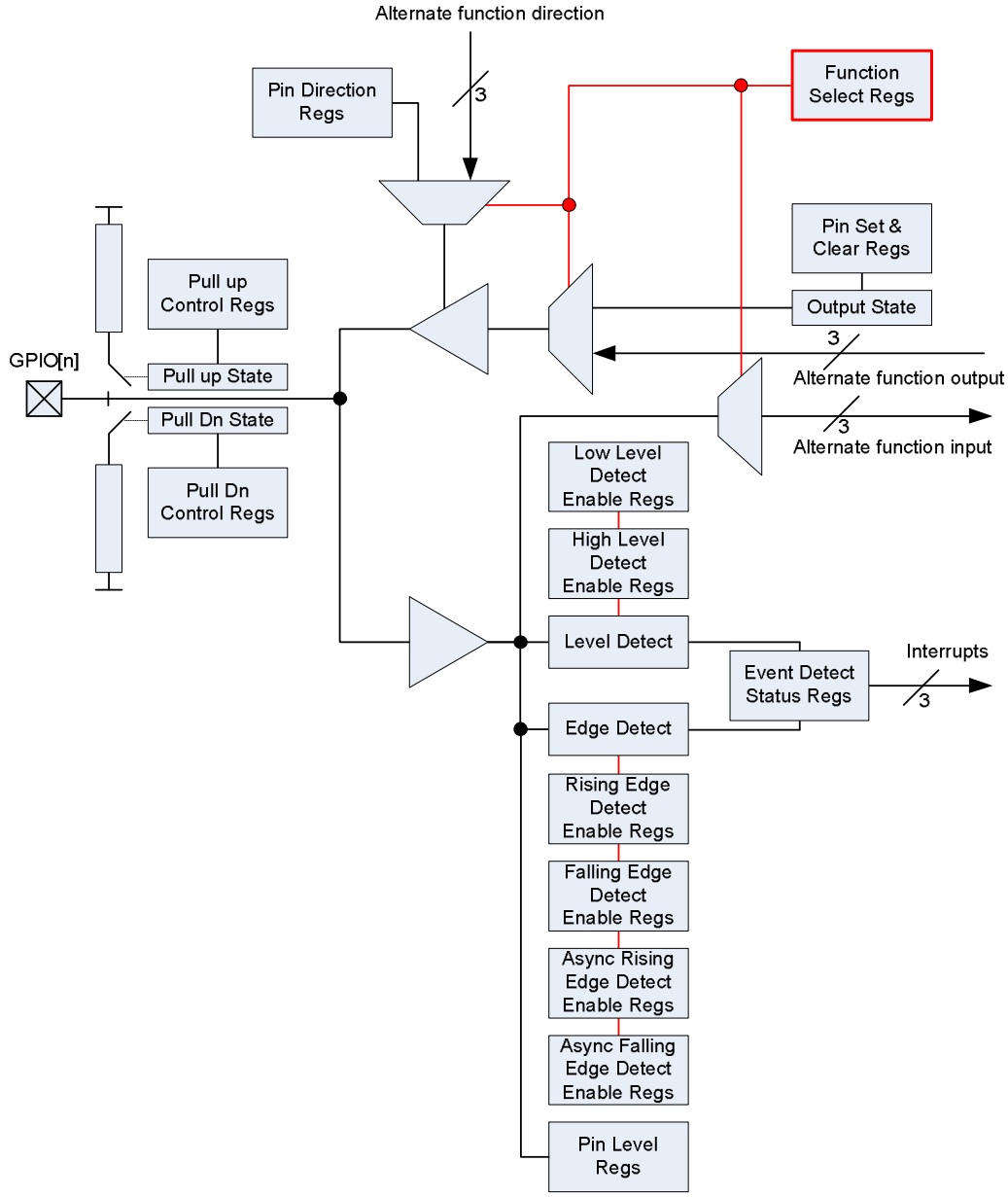


Figure 6-1 GPIO Block Diagram

The GPIO peripheral has three dedicated interrupt lines. These lines are triggered by the setting of bits in the event detect status register. Each bank has its' own interrupt line with the third line shared between all bits.

The Alternate function table also has the pull state (pull-up/pull-down) which is applied after a power down.

6.1 Register View

The GPIO has 41 registers. All accesses are assumed to be 32-bit.

Address	Field Name	Description	Size	Read/ Write
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0000	GPFSEL0	GPIO Function Select 0	32	R/W
0x 7E20 0004	GPFSEL1	GPIO Function Select 1	32	R/W
0x 7E20 0008	GPFSEL2	GPIO Function Select 2	32	R/W
0x 7E20 000C	GPFSEL3	GPIO Function Select 3	32	R/W
0x 7E20 0010	GPFSEL4	GPIO Function Select 4	32	R/W
0x 7E20 0014	GPFSEL5	GPIO Function Select 5	32	R/W
0x 7E20 0018	-	Reserved	-	-
0x 7E20 001C	GPSET0	GPIO Pin Output Set 0	32	W
0x 7E20 0020	GPSET1	GPIO Pin Output Set 1	32	W
0x 7E20 0024	-	Reserved	-	-
0x 7E20 0028	GPCLR0	GPIO Pin Output Clear 0	32	W
0x 7E20 002C	GPCLR1	GPIO Pin Output Clear 1	32	W
0x 7E20 0030	-	Reserved	-	-
0x 7E20 0034	GPLEV0	GPIO Pin Level 0	32	R
0x 7E20 0038	GPLEV1	GPIO Pin Level 1	32	R
0x 7E20 003C	-	Reserved	-	-
0x 7E20 0040	GPEDS0	GPIO Pin Event Detect Status 0	32	R/W
0x 7E20 0044	GPEDS1	GPIO Pin Event Detect Status 1	32	R/W
0x 7E20 0048	-	Reserved	-	-
0x 7E20 004C	GPREN0	GPIO Pin Rising Edge Detect Enable 0	32	R/W
0x 7E20 0050	GPREN1	GPIO Pin Rising Edge Detect Enable 1	32	R/W
0x 7E20 0054	-	Reserved	-	-
0x 7E20 0058	GPFEN0	GPIO Pin Falling Edge Detect Enable 0	32	R/W
0x 7E20 005C	GPFEN1	GPIO Pin Falling Edge Detect Enable 1	32	R/W

Address	Field Name	Description	Size	Read/ Write
0x 7E20 0060	-	Reserved	-	-
0x 7E20 0064	GPHEN0	GPIO Pin High Detect Enable 0	32	R/W
0x 7E20 0068	GPHEN1	GPIO Pin High Detect Enable 1	32	R/W
0x 7E20 006C	-	Reserved	-	-
0x 7E20 0070	GPLEN0	GPIO Pin Low Detect Enable 0	32	R/W
0x 7E20 0074	GPLEN1	GPIO Pin Low Detect Enable 1	32	R/W
0x 7E20 0078	-	Reserved	-	-
0x 7E20 007C	GPAREN0	GPIO Pin Async. Rising Edge Detect 0	32	R/W
0x 7E20 0080	GPAREN1	GPIO Pin Async. Rising Edge Detect 1	32	R/W
0x 7E20 0084	-	Reserved	-	-
0x 7E20 0088	GPAFEN0	GPIO Pin Async. Falling Edge Detect 0	32	R/W
0x 7E20 008C	GPAFEN1	GPIO Pin Async. Falling Edge Detect 1	32	R/W
0x 7E20 0090	-	Reserved	-	-
0x 7E20 0094	GPPUD	GPIO Pin Pull-up/down Enable	32	R/W
0x 7E20 0098	GPPUDCLK0	GPIO Pin Pull-up/down Enable Clock 0	32	R/W
0x 7E20 009C	GPPUDCLK1	GPIO Pin Pull-up/down Enable Clock 1	32	R/W
0x 7E20 00A0	-	Reserved	-	-
0x 7E20 00B0	-	Test	4	R/W

Table 6-1 GPIO Register Assignment

GPIO Function Select Registers (GPFSELn)

SYNOPSIS The function select registers are used to define the operation of the general-purpose I/O pins. Each of the 54 GPIO pins has at least two alternative functions as defined in section 16.2. The FSEL{n} field determines the functionality of the nth GPIO pin. All unused alternative function lines are tied to ground and will output a “0” if selected. All pins reset to normal GPIO input operation.

Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0

29-27	FSEL9	FSEL9 - Function Select 9 000 = GPIO Pin 9 is an input 001 = GPIO Pin 9 is an output 100 = GPIO Pin 9 takes alternate function 0 101 = GPIO Pin 9 takes alternate function 1 110 = GPIO Pin 9 takes alternate function 2 111 = GPIO Pin 9 takes alternate function 3 011 = GPIO Pin 9 takes alternate function 4 010 = GPIO Pin 9 takes alternate function 5	R/W	0
26-24	FSEL8	FSEL8 - Function Select 8	R/W	0
23-21	FSEL7	FSEL7 - Function Select 7	R/W	0
20-18	FSEL6	FSEL6 - Function Select 6	R/W	0
17-15	FSEL5	FSEL5 - Function Select 5	R/W	0
14-12	FSEL4	FSEL4 - Function Select 4	R/W	0
11-9	FSEL3	FSEL3 - Function Select 3	R/W	0
8-6	FSEL2	FSEL2 - Function Select 2	R/W	0
5-3	FSEL1	FSEL1 - Function Select 1	R/W	0
2-0	FSEL0	FSEL0 - Function Select 0	R/W	0

Table 6-2 – GPIO Alternate function select register 0

Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL19	FSEL19 - Function Select 19 000 = GPIO Pin 19 is an input 001 = GPIO Pin 19 is an output 100 = GPIO Pin 19 takes alternate function 0 101 = GPIO Pin 19 takes alternate function 1 110 = GPIO Pin 19 takes alternate function 2 111 = GPIO Pin 19 takes alternate function 3 011 = GPIO Pin 19 takes alternate function 4 010 = GPIO Pin 19 takes alternate function 5	R/W	0
26-24	FSEL18	FSEL18 - Function Select 18	R/W	0
23-21	FSEL17	FSEL17 - Function Select 17	R/W	0
20-18	FSEL16	FSEL16 - Function Select 16	R/W	0
17-15	FSEL15	FSEL15 - Function Select 15	R/W	0
14-12	FSEL14	FSEL14 - Function Select 14	R/W	0
11-9	FSEL13	FSEL13 - Function Select 13	R/W	0
8-6	FSEL12	FSEL12 - Function Select 12	R/W	0
5-3	FSEL11	FSEL11 - Function Select 11	R/W	0
2-0	FSEL10	FSEL10 - Function Select 10	R/W	0

Table 6-3 – GPIO Alternate function select register 1

Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL29	FSEL29 - Function Select 29 000 = GPIO Pin 29 is an input 001 = GPIO Pin 29 is an output 100 = GPIO Pin 29 takes alternate function 0 101 = GPIO Pin 29 takes alternate function 1 110 = GPIO Pin 29 takes alternate function 2 111 = GPIO Pin 29 takes alternate function 3 011 = GPIO Pin 29 takes alternate function 4 010 = GPIO Pin 29 takes alternate function 5	R/W	0
26-24	FSEL28	FSEL28 - Function Select 28	R/W	0
23-21	FSEL27	FSEL27 - Function Select 27	R/W	0
20-18	FSEL26	FSEL26 - Function Select 26	R/W	0
17-15	FSEL25	FSEL25 - Function Select 25	R/W	0
14-12	FSEL24	FSEL24 - Function Select 24	R/W	0
11-9	FSEL23	FSEL23 - Function Select 23	R/W	0
8-6	FSEL22	FSEL22 - Function Select 22	R/W	0
5-3	FSEL21	FSEL21 - Function Select 21	R/W	0
2-0	FSEL20	FSEL20 - Function Select 20	R/W	0

Table 6-4 – GPIO Alternate function select register 2

Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL39	FSEL39 - Function Select 39 000 = GPIO Pin 39 is an input 001 = GPIO Pin 39 is an output 100 = GPIO Pin 39 takes alternate function 0 101 = GPIO Pin 39 takes alternate function 1 110 = GPIO Pin 39 takes alternate function 2 111 = GPIO Pin 39 takes alternate function 3 011 = GPIO Pin 39 takes alternate function 4 010 = GPIO Pin 39 takes alternate function 5	R/W	0
26-24	FSEL38	FSEL38 - Function Select 38	R/W	0
23-21	FSEL37	FSEL37 - Function Select 37	R/W	0
20-18	FSEL36	FSEL36 - Function Select 36	R/W	0
17-15	FSEL35	FSEL35 - Function Select 35	R/W	0
14-12	FSEL34	FSEL34 - Function Select 34	R/W	0
11-9	FSEL33	FSEL33 - Function Select 33	R/W	0
8-6	FSEL32	FSEL32 - Function Select 32	R/W	0

5-3	FSEL31	FSEL31 - Function Select 31	R/W	0
2-0	FSEL30	FSEL30 - Function Select 30	R/W	0

Table 6-5 – GPIO Alternate function select register 3

Bit(s)	Field Name	Description	Type	Reset
31-30	---	Reserved	R	0
29-27	FSEL49	FSEL49 - Function Select 49 000 = GPIO Pin 49 is an input 001 = GPIO Pin 49 is an output 100 = GPIO Pin 49 takes alternate function 0 101 = GPIO Pin 49 takes alternate function 1 110 = GPIO Pin 49 takes alternate function 2 111 = GPIO Pin 49 takes alternate function 3 011 = GPIO Pin 49 takes alternate function 4 010 = GPIO Pin 49 takes alternate function 5	R/W	0
26-24	FSEL48	FSEL48 - Function Select 48	R/W	0
23-21	FSEL47	FSEL47 - Function Select 47	R/W	0
20-18	FSEL46	FSEL46 - Function Select 46	R/W	0
17-15	FSEL45	FSEL45 - Function Select 45	R/W	0
14-12	FSEL44	FSEL44 - Function Select 44	R/W	0
11-9	FSEL43	FSEL43 - Function Select 43	R/W	0
8-6	FSEL42	FSEL42 - Function Select 42	R/W	0
5-3	FSEL41	FSEL41 - Function Select 41	R/W	0
2-0	FSEL40	FSEL40 - Function Select 40	R/W	0

Table 6-6 – GPIO Alternate function select register 4

Bit(s)	Field Name	Description	Type	Reset
31-12	---	Reserved	R	0
11-9	FSEL53	FSEL53 - Function Select 53 000 = GPIO Pin 53 is an input 001 = GPIO Pin 53 is an output 100 = GPIO Pin 53 takes alternate function 0 101 = GPIO Pin 53 takes alternate function 1 110 = GPIO Pin 53 takes alternate function 2 111 = GPIO Pin 53 takes alternate function 3 011 = GPIO Pin 53 takes alternate function 4 010 = GPIO Pin 53 takes alternate function 5	R/W	0
8-6	FSEL52	FSEL52 - Function Select 52	R/W	0
5-3	FSEL51	FSEL51 - Function Select 51	R/W	0
2-0	FSEL50	FSEL50 - Function Select 50	R/W	0

Table 6-7 – GPIO Alternate function select register 5

GPIO Pin Output Set Registers (GPSETn)

SYNOPSIS The output set registers are used to set a GPIO pin. The SET{n} field defines the respective GPIO pin to set, writing a "0" to the field has no effect. If the GPIO pin is being used as in input (by default) then the value in the SET{n} field is ignored. However, if the pin is subsequently defined as an output then the bit will be set according to the last set/clear operation. Separating the set and clear functions removes the need for read-modify-write operations

Bit(s)	Field Name	Description	Type	Reset
31-0	SETn (n=0..31)	0 = No effect 1 = Set GPIO pin *n*	R/W	0

Table 6-8 – GPIO Output Set Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	SETn (n=32..53)	0 = No effect 1 = Set GPIO pin *n.*	R/W	0

Table 6-9 – GPIO Output Set Register 1

GPIO Pin Output Clear Registers (GPCLRn)

SYNOPSIS The output clear registers) are used to clear a GPIO pin. The CLR{n} field defines the respective GPIO pin to clear, writing a "0" to the field has no effect. If the GPIO pin is being used as in input (by default) then the value in the CLR{n} field is ignored. However, if the pin is subsequently defined as an output then the bit will be set according to the last set/clear operation. Separating the set and clear functions removes the need for read-modify-write operations.

Bit(s)	Field Name	Description	Type	Reset
31-0	CLRn (n=0..31)	0 = No effect 1 = Clear GPIO pin *n*	R/W	0

Table 6-10 – GPIO Output Clear Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0

21-0	CLRn (n=32..53)	0 = No effect 1 = Set GPIO pin *n*	R/W	0

Table 6-11 – GPIO Output Clear Register 1

GPIO Pin Level Registers (GPLEVn)

SYNOPSIS The pin level registers return the actual value of the pin. The LEV{n} field gives the value of the respective GPIO pin.

Bit(s)	Field Name	Description	Type	Reset
31-0	LEVn (n=0..31)	0 = GPIO pin *n* is low 0 = GPIO pin *n* is high	R/W	0

Table 6-12 – GPIO Level Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	LEVn (n=32..53)	0 = GPIO pin *n* is low 0 = GPIO pin *n* is high	R/W	0

Table 6-13 – GPIO Level Register 1

GPIO Event Detect Status Registers (GPEDSn)

SYNOPSIS The event detect status registers are used to record level and edge events on the GPIO pins. The relevant bit in the event detect status registers is set whenever: 1) an edge is detected that matches the type of edge programmed in the rising/falling edge detect enable registers, or 2) a level is detected that matches the type of level programmed in the high/low level detect enable registers. The bit is cleared by writing a "1" to the relevant bit.

The interrupt controller can be programmed to interrupt the processor when any of the status bits are set. The GPIO peripheral has three dedicated interrupt lines. Each GPIO bank can generate an independent interrupt. The third line generates a single interrupt whenever any bit is set.

Bit(s)	Field Name	Description	Type	Reset
31-0	EDSn (n=0..31)	0 = Event not detected on GPIO pin n 1 = Event detected on GPIO pin n	R/W	0

Table 6-14 – GPIO Event Detect Status Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	EDSn (n=32..53)	0 = Event not detected on GPIO pin n 1 = Event detected on GPIO pin n	R/W	0

Table 6-15 – GPIO Event Detect Status Register 1

GPIO Rising Edge Detect Enable Registers (GPRENn)

SYNOPSIS The rising edge detect enable registers define the pins for which a rising edge transition sets a bit in the event detect status registers (GPEDSn). When the relevant bits are set in both the GPRENn and GPFENn registers, any transition (1 to 0 and 0 to 1) will set a bit in the GPEDSn registers. The GPRENn registers use synchronous edge detection. This means the input signal is sampled using the system clock and then it is looking for a "011" pattern on the sampled signal. This has the effect of suppressing glitches.

Bit(s)	Field Name	Description	Type	Reset
31-0	RENn (n=0..31)	0 = Rising edge detect disabled on GPIO pin *n.* 1 = Rising edge on GPIO pin *n* sets corresponding bit in EDSn.	R/W	0

Table 6-16 – GPIO Rising Edge Detect Status Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	RENn (n=32..53)	0 = Rising edge detect disabled on GPIO pin *n.* 1 = Rising edge on GPIO pin *n* sets corresponding bit in EDSn.	R/W	0

Table 6-17 – GPIO Rising Edge Detect Status Register 1

GPIO Falling Edge Detect Enable Registers (GPRENn)

SYNOPSIS The falling edge detect enable registers define the pins for which a falling edge transition sets a bit in the event detect status registers (GPEDSn). When the relevant bits are set in both the GPRENn and GPFENn registers, any transition (1 to 0 and 0 to 1) will set a bit in the GPEDSn registers. The GPFENn registers use synchronous edge detection. This means the input signal is sampled using the system clock and then it is looking for a "100" pattern on the sampled signal. This has the effect of suppressing glitches.

Bit(s)	Field Name	Description	Type	Reset
31-0	FENn (n=0..31)	0 = Falling edge detect disabled on GPIO pin *n.* 1 = Falling edge on GPIO pin *n* sets corresponding bit in EDSn.	R/W	0

Table 6-18 – GPIO Falling Edge Detect Status Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	FENn (n=32..53)	0 = Falling edge detect disabled on GPIO pin *n.* 1 = Falling edge on GPIO pin *n* sets corresponding bit in EDSn.	R/W	0

Table 6-19 – GPIO Falling Edge Detect Status Register 1

GPIO High Detect Enable Registers (GPHENn)

SYNOPSIS The high level detect enable registers define the pins for which a high level sets a bit in the event detect status register (GPEDSn). If the pin is still high when an attempt is made to clear the status bit in GPEDSn then the status bit will remain set.

Bit(s)	Field Name	Description	Type	Reset
31-0	HENn (n=0..31)	0 = High detect disabled on GPIO pin *n* 1 = High on GPIO pin *n* sets corresponding bit in GPEDS	R/W	0

Table 6-20 – GPIO High Detect Status Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	HENn (n=32..53)	0 = High detect disabled on GPIO pin *n* 1 = High on GPIO pin *n* sets corresponding bit in GPEDS	R/W	0

Table 6-21 – GPIO High Detect Status Register 1

GPIO Low Detect Enable Registers (GPLENn)

SYNOPSIS The low level detect enable registers define the pins for which a low level sets a bit in the event detect status register (GPEDSn). If the pin is still low when an attempt is made to clear the status bit in GPEDSn then the status bit will remain set.

Bit(s)	Field Name	Description	Type	Reset
31-0	LENn (n=0..31)	0 = Low detect disabled on GPIO pin *n* 1 = Low on GPIO pin *n* sets corresponding bit in GPEDS	R/W	0

Table 6-22 – GPIO Low Detect Status Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	LENn (n=32..53)	0 = Low detect disabled on GPIO pin *n* 1 = Low on GPIO pin *n* sets corresponding bit in GPEDS	R/W	0

Table 6-23 – GPIO Low Detect Status Register 1

GPIO Asynchronous rising Edge Detect Enable Registers (GPARENn)

SYNOPSIS The asynchronous rising edge detect enable registers define the pins for which a asynchronous rising edge transition sets a bit in the event detect status registers (GPEDSn).

Asynchronous means the incoming signal is not sampled by the system clock. As such rising edges of very short duration can be detected.

Bit(s)	Field Name	Description	Type	Reset
31-0	ARENn (n=0..31)	0 = Asynchronous rising edge detect disabled on GPIO pin *n.* 1 = Asynchronous rising edge on GPIO pin *n* sets corresponding bit in EDSn.	R/W	0

Table 6-24 – GPIO Asynchronous rising Edge Detect Status Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	ARENn (n=32..53)	0 = Asynchronous rising edge detect disabled on GPIO pin *n.* 1 = Asynchronous rising edge on GPIO pin *n* sets corresponding bit in EDSn.	R/W	0

Table 6-25 – GPIO Asynchronous rising Edge Detect Status Register 1

GPIO Asynchronous Falling Edge Detect Enable Registers (GPAFENn)

SYNOPSIS The asynchronous falling edge detect enable registers define the pins for which a asynchronous falling edge transition sets a bit in the event detect status registers (GPEDSn). Asynchronous means the incoming signal is not sampled by the system clock. As such falling edges of very short duration can be detected.

Bit(s)	Field Name	Description	Type	Reset
31-0	AFENn (n=0..31)	0 = Asynchronous falling edge detect disabled on GPIO pin *n.* 1 = Asynchronous falling edge on GPIO pin *n* sets corresponding bit in EDSn.	R/W	0

Table 6-26 – GPIO Asynchronous Falling Edge Detect Status Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	AFENn (n=32..53)	0 = Asynchronous falling edge detect disabled on GPIO pin *n.* 1 = Asynchronous falling edge on GPIO pin *n* sets corresponding bit in EDSn.	R/W	0

Table 6-27 – GPIO Asynchronous Falling Edge Detect Status Register 1

GPIO Pull-up/down Register (GPPUD)

SYNOPSIS The GPIO Pull-up/down Register controls the actuation of the internal pull-up/down control line to ALL the GPIO pins. This register must be used in conjunction with the 2 GPPUDCLKn registers.

Note that it is not possible to read back the current Pull-up/down settings and so it is the users' responsibility to 'remember' which pull-up/downs are active. The reason for this is that GPIO pull-ups are maintained even in power-down mode when the core is off, when all register contents is lost.

The Alternate function table also has the pull state which is applied after a power down.

Bit(s)	Field Name	Description	Type	Reset

31-2	---	Unused	R	0
1-0	PUD	PUD - GPIO Pin Pull-up/down 00 = Off – disable pull-up/down 01 = Enable Pull Down control 10 = Enable Pull Up control 11 = Reserved *Use in conjunction with GPPUDCLK0/1/2	R/W	0

Table 6-28 – GPIO Pull-up/down Register (GPPUD)

GPIO Pull-up/down Clock Registers (GPPUDCLKn)

SYNOPSIS

The GPIO Pull-up/down Clock Registers control the actuation of internal pull-downs on the respective GPIO pins. These registers must be used in conjunction with the GPPUD register to effect GPIO Pull-up/down changes. The following sequence of events is required:

1. Write to GPPUD to set the required control signal (i.e. Pull-up or Pull-Down or neither to remove the current Pull-up/down)
2. Wait 150 cycles – this provides the required set-up time for the control signal
3. Write to GPPUDCLK0/1 to clock the control signal into the GPIO pads you wish to modify – NOTE only the pads which receive a clock will be modified, all others will retain their previous state.
4. Wait 150 cycles – this provides the required hold time for the control signal
5. Write to GPPUD to remove the control signal
6. Write to GPPUDCLK0/1 to remove the clock

Bit(s)	Field Name	Description	Type	Reset
(31-0)	PUDCLKn (n=0..31)	0 = No Effect 1 = Assert Clock on line *(n)* *Must be used in conjunction with GPPUD	R/W	0

Table 6-29 – GPIO Pull-up/down Clock Register 0

Bit(s)	Field Name	Description	Type	Reset
31-22	-	Reserved	R	0
21-0	PUDCLKn (n=32..53)	0 = No Effect 1 = Assert Clock on line *(n)* *Must be used in conjunction with GPPUD	R/W	0

Table 6-30 – GPIO Pull-up/down Clock Register 1

6.2 Alternative Function Assignments

Every GPIO pin can carry an alternate function. Up to 6 alternate function are available but not every pin has that many alternate functions. The table below gives a quick over view.

	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO0	High	SDA0	SA5	<reserved>			
GPIO1	High	SCL0	SA4	<reserved>			
GPIO2	High	SDA1	SA3	<reserved>			
GPIO3	High	SCL1	SA2	<reserved>			
GPIO4	High	GPCLK0	SA1	<reserved>			ARM_TDI
GPIO5	High	GPCLK1	SA0	<reserved>			ARM_TDO
GPIO6	High	GPCLK2	SOE_N / SE	<reserved>			ARM_RTCK
GPIO7	High	SPI0_CE1_N	SWE_N / SRW_N	<reserved>			
GPIO8	High	SPI0_CE0_N	SD0	<reserved>			
GPIO9	Low	SPI0_MISO	SD1	<reserved>			
GPIO10	Low	SPI0_MOSI	SD2	<reserved>			
GPIO11	Low	SPI0_SCLK	SD3	<reserved>			
GPIO12	Low	PWM0	SD4	<reserved>			ARM_TMS
GPIO13	Low	PWM1	SD5	<reserved>			ARM_TCK
GPIO14	Low	TXD0	SD6	<reserved>			TXD1
GPIO15	Low	RXD0	SD7	<reserved>			RXD1
GPIO16	Low	<reserved>	SD8	<reserved>	CTS0	SPI1_CE2_N	CTS1
GPIO17	Low	<reserved>	SD9	<reserved>	RTS0	SPI1_CE1_N	RTS1
GPIO18	Low	PCM_CLK	SD10	<reserved>	BSCSL SDA / MOSI	SPI1_CE0_N	PWM0
GPIO19	Low	PCM_FS	SD11	<reserved>	BSCSL SCL / SCLK	SPI1_MISO	PWM1
GPIO20	Low	PCM_DIN	SD12	<reserved>	BSCSL / MISO	SPI1_MOSI	GPCLK0
GPIO21	Low	PCM_DOUT	SD13	<reserved>	BSCSL / CE_N	SPI1_SCLK	GPCLK1
GPIO22	Low	<reserved>	SD14	<reserved>	SD1_CLK	ARM_TRST	
GPIO23	Low	<reserved>	SD15	<reserved>	SD1_CMD	ARM_RTCK	
GPIO24	Low	<reserved>	SD16	<reserved>	SD1_DAT0	ARM_TDO	
GPIO25	Low	<reserved>	SD17	<reserved>	SD1_DAT1	ARM_TCK	
GPIO26	Low	<reserved>	<reserved>	<reserved>	SD1_DAT2	ARM_TDI	
GPIO27	Low	<reserved>	<reserved>	<reserved>	SD1_DAT3	ARM_TMS	
GPIO28	-	SDA0	SA5	PCM_CLK	<reserved>		
GPIO29	-	SCL0	SA4	PCM_FS	<reserved>		
GPIO30	Low	<reserved>	SA3	PCM_DIN	CTS0		CTS1
GPIO31	Low	<reserved>	SA2	PCM_DOUT	RTS0		RTS1
GPIO32	Low	GPCLK0	SA1	<reserved>	TXD0		TXD1
GPIO33	Low	<reserved>	SA0	<reserved>	RXD0		RXD1
GPIO34	High	GPCLK0	SOE_N / SE	<reserved>	<reserved>		
GPIO35	High	SPI0_CE1_N	SWE_N / SRW_N		<reserved>		
GPIO36	High	SPI0_CE0_N	SD0	TXD0	<reserved>		
GPIO37	Low	SPI0_MISO	SD1	RXD0	<reserved>		
GPIO38	Low	SPI0_MOSI	SD2	RTS0	<reserved>		
GPIO39	Low	SPI0_SCLK	SD3	CTS0	<reserved>		
GPIO40	Low	PWM0	SD4		<reserved>	SPI2_MISO	TXD1
	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5

GPIO41	Low	PWM1	SD5	<reserved>	<reserved>	SPI2_MOSI	RXD1
GPIO42	Low	GPCLK1	SD6	<reserved>	<reserved>	SPI2_SCLK	RTS1
GPIO43	Low	GPCLK2	SD7	<reserved>	<reserved>	SPI2_CE0_N	CTS1
GPIO44	-	GPCLK1	SDA0	SDA1	<reserved>	SPI2_CE1_N	
GPIO45	-	PWM1	SCL0	SCL1	<reserved>	SPI2_CE2_N	
GPIO46	High	<Internal>					
GPIO47	High	<Internal>					
GPIO48	High	<Internal>					
GPIO49	High	<Internal>					
GPIO50	High	<Internal>					
GPIO51	High	<Internal>					
GPIO52	High	<Internal>					
GPIO53	High	<Internal>					

Table 6-31 GPIO Pins Alternative Function Assignment

Entries which are white should ***not*** be used. These may have unexpected results as some of these have special functions used in test mode. e.g. they may drive the output with high frequency signals.

Special function legend:

Name	Function	See section
SDA0	BSC[6] master 0 data line	BSC
SCL0	BSC master 0 clock line	BSC
SDA1	BSC master 1 data line	BSC
SCL1	BSC master 1 clock line	BSC
GPCLK0	General purpose Clock 0	**<TBD>**
GPCLK1	General purpose Clock 1	**<TBD>**
GPCLK2	General purpose Clock 2	**<TBD>**
SPI0_CE1_N	SPI0 Chip select 1	SPI
SPI0_CE0_N	SPI0 Chip select 0	SPI
SPI0_MISO	SPI0 MISO	SPI
SPI0_MOSI	SPI0 MOSI	SPI
SPI0_SCLK	SPI0 Serial clock	SPI
PWMx	Pulse Width Modulator 0..1	Pulse Width Modulator
TXD0	UART 0 Transmit Data	UART
RXD0	UART 0 Receive Data	UART
CTS0	UART 0 Clear To Send	UART
RTS0	UART 0 Request To Send	UART
PCM_CLK	PCM clock	PCM Audio
PCM_FS	PCM Frame Sync	PCM Audio
PCM_DIN	PCM Data in	PCM Audio
PCM_DOUT	PCM data out	PCM Audio
SAx	Secondary mem Address bus	Secondary Memory Interface
SOE_N / SE	Secondary mem. Controls	Secondary Memory Interface
SWE_N / SRW_N	Secondary mem. Controls	Secondary Memory Interface
SDx	Secondary mem. data bus	Secondary Memory Interface
BSCSL SDA / MOSI	BSC slave Data, SPI salve MOSI	BSC ISP slave
BSCSL SCL / SCLK	BSC slave Clock, SPI slave clock	BSC ISP slave
BSCSL - / MISO	BSC <not used>,SPI MISO	BSC ISP slave
BSCSL - / CE_N	BSC <not used>, SPI CSn	BSC ISP slave

[6] The Broadcom Serial Control bus is a proprietary bus compliant with the Philips® I2C bus/interface

Name	Function	See section
SPI1_CEx_N	SPI1 Chip select 0-2	Auxiliary I/O
SPI1_MISO	SPI1 MISO	Auxiliary I/O
SPI1_MOSI	SPI1 MOSI	Auxiliary I/O
SPI1_SCLK	SPI1 Serial clock	Auxiliary I/O
TXD0	UART 1 Transmit Data	Auxiliary I/O
RXD0	UART 1 Receive Data	Auxiliary I/O
CTS0	UART 1 Clear To Send	Auxiliary I/O
RTS0	UART 1 Request To Send	Auxiliary I/O
SPI2_CEx_N	SPI2 Chip select 0-2	Auxiliary I/O
SPI2_MISO	SPI2 MISO	Auxiliary I/O
SPI2_MOSI	SPI2 MOSI	Auxiliary I/O
SPI2_SCLK	SPI2 Serial clock	Auxiliary I/O
ARM_TRST	ARM JTAG reset	**<TBD>**
ARM_RTCK	ARM JTAG return clock	**<TBD>**
ARM_TDO	ARM JTAG Data out	**<TBD>**
ARM_TCK	ARM JTAG Clock	**<TBD>**
ARM_TDI	ARM JTAG Data in	**<TBD>**
ARM_TMS	ARM JTAG Mode select	**<TBD>**

6.3 General Purpose GPIO Clocks

The General Purpose clocks can be output to GPIO pins. They run from the peripherals clock sources and use clock generators with noise-shaping MASH dividers. These allow the GPIO clocks to be used to drive audio devices.

The fractional divider operates by periodically dropping source clock pulses, therefore the output frequency will periodically switch between:

$$\frac{source\_frequency}{DIVI} \quad \& \quad \frac{source\_frequency}{DIVI+1}$$

Jitter is therefore reduced by increasing the source clock frequency. In applications where jitter is a concern, the fastest available clock source should be used.

The General Purpose clocks have MASH noise-shaping dividers which push this fractional divider jitter out of the audio band.

MASH noise-shaping is incorporated to push the fractional divider jitter out of the audio band if required. The MASH can be programmed for 1, 2 or 3-stage filtering. MASH filter, the frequency is spread around the requested frequency and the user must ensure that the module is not exposed to frequencies higher than 25MHz. Also, the MASH filter imposes a low limit on the range of DIVI.

MASH	min DIVI	min output freq	average output freq	max output freq
0 (int divide)	1	source / (DIVI)	source / (DIVI)	source / (DIVI)
1	2	source / (DIVI)	source / (DIVI + DIVF / 1024)	source / (DIVI + 1)
2	3	source / (DIVI - 1)	source / (DIVI + DIVF / 1024)	source / (DIVI + 2)
3	5	source / (DIVI - 3)	source / (DIVI + DIVF / 1024)	source / (DIVI + 4)

Table 6-32 Effect of MASH Filter on Frequency

The following example illustrates the spreading of output clock frequency resulting from the use of the MASH filter. Note that the spread is greater for lower divisors.

PLL freq (MHz)	target freq (MHz)	MASH	divisor	DIVI	DIVF	min freq (MHz)	ave freq (MHz)	max freq (MHz)	error
650	18.32	0	35.480	35	492	18.57	18.57	18.57	ok
650	18.32	1	35.480	35	492	18.06	18.32	18.57	ok
650	18.32	2	35.480	35	492	17.57	18.32	19.12	ok
650	18.32	3	35.480	35	492	16.67	18.32	20.31	ok
400	18.32	0	21.834	21	854	19.05	19.05	19.05	ok
400	18.32	1	21.834	21	854	18.18	18.32	19.05	ok
400	18.32	2	21.834	21	854	17.39	18.32	20.00	ok
400	18.32	3	21.834	21	854	16.00	18.32	22.22	ok
200	18.32	0	10.917	10	939	20.00	20.00	20.00	ok
200	18.32	1	10.917	10	939	18.18	18.32	20.00	ok
200	18.32	2	10.917	10	939	16.67	18.32	22.22	ok
200	18.32	3	10.917	10	939	14.29	18.32	28.57	error

Table 6-33 Example of Frequency Spread when using MASH Filtering

It is beyond the scope of this specification to describe the operation of a MASH filter or to determine under what conditions the available levels of filtering are beneficial.

Operating Frequency

The maximum operating frequency of the General Purpose clocks is ~125MHz at 1.2V but this will be reduced if the GPIO pins are heavily loaded or have a capacitive load.

Clock Manager General Purpose Clocks Control (CM_GP0CTL, GP1CTL & GP2CTL)

Address 0x 7e10 1070 CM_GP0CTL
0x 7e10 1078 CM_GP1CTL
0x 7e10 1080 CM_GP2CTL

Bit Number	Field Name	Description	Read/ Write	Reset
31-24	PASSWD	Clock Manager password “5a”	W	0
23-11	-	Unused	R	0
10-9	MASH	MASH control 0 = integer division 1 = 1-stage MASH (equivalent to non-MASH dividers) 2 = 2-stage MASH 3 = 3-stage MASH To avoid lock-ups and glitches do not change this control while BUSY=1 and do not change this control at the same time as asserting ENAB.	R/W	0
8	FLIP	Invert the clock generator output This is intended for use in test/debug only. Switching this control will generate an edge on the clock generator output. To avoid output glitches do not switch this control while BUSY=1.	R/W	0
7	BUSY	Clock generator is running Indicates the clock generator is running. To avoid glitches and lock-ups, clock sources and setups must not be changed while this flag is set.	R	0
6	-	Unused	R	0
5	KILL	Kill the clock generator 0 = no action 1 = stop and reset the clock generator This is intended for test/debug only. Using this control may cause a glitch on the clock generator output.	R/W	0
4	ENAB	Enable the clock generator This requests the clock to start or stop without glitches. The output clock will not stop immediately because the cycle must be allowed to complete to avoid glitches. The BUSY flag will go low when the final cycle is completed.	R/W	0
3-0	SRC	Clock source 0 = GND 1 = oscillator 2 = testdebug0 3 = testdebug1 4 = PLLA per 5 = PLLC per 6 = PLLD per 7 = HDMI auxiliary 8-15 = GND To avoid lock-ups and glitches do not change this control while BUSY=1 and do not change this control at the same time as asserting ENAB.	R/W	0

Table 6-34 General Purpose Clocks Control

Clock Manager General Purpose Clock Divisors (CM_GP0DIV, CM_GP1DIV & CM_GP2DIV)

Address 0x 7e10 1074 CM_GP0DIV
0x 7e10 107c CM_GP1DIV
0x 7e10 1084 CM_GP2DIV

Bit Number	Field Name	Description	Read/ Write	Reset
31-24	PASSWD	Clock Manager password “5a”	W	0
23-12	DIVI	Integer part of divisor This value has a minimum limit determined by the MASH setting. See text for details. To avoid lock-ups and glitches do not change this control while BUSY=1.	R/W	0
11-0	DIVF	Fractional part of divisor To avoid lock-ups and glitches do not change this control while BUSY=1.	R/W	0

Table 6-35 General Purpose Clock Divisors

7 Interrupts

7.1 Introduction

The ARM has two types of interrupt sources:

1. Interrupts coming from the GPU peripherals.
2. Interrupts coming from local ARM control peripherals.

The ARM processor gets three types of interrupts:

1. Interrupts from ARM specific peripherals.
2. Interrupts from GPU peripherals.
3. Special events interrupts.

The ARM specific interrupts are:

- One timer.
- One Mailbox.
- Two Doorbells.
- Two GPU halted interrupts.
- Two Address/access error interrupt

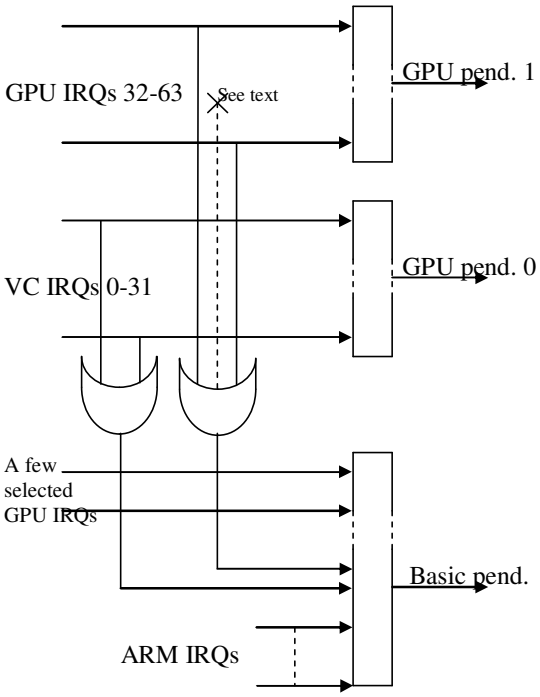
The Mailbox and Doorbell registers are not for general usage.

For each interrupt source (ARM or GPU) there is an interrupt enable bit (read/write) and an interrupt pending bit (Read Only). All interrupts generated by the arm control block are level sensitive interrupts. Thus all interrupts remain asserted until disabled or the interrupt source is cleared.

Default the interrupts from doorbell 0,1 and mailbox 0 go to the ARM this means that these resources should be written by the GPU and read by the ARM. The opposite holds for doorbells 2, 3 and mailbox 1.

7.2 Interrupt pending.

An interrupt vector module has NOT been implemented. To still have adequate interrupt processing the interrupt pending bits are organized as follows:



There are three interrupt pending registers.

One basic pending register and two GPU pending registers.

Basic pending register.

The basic pending register has interrupt pending bits for the ARM specific interrupts .

To speed up the interrupt processing it also has a number of selected GPU interrupts which are deemed most likely to be required in ARM drivers.

Further there are two special GPU pending bits which tell if any of the two other pending registers has bits set, one bit if a GPU interrupt 0-31 is pending, a second bit if a GPU interrupt 32-63 is pending. The 'selected GPU interrupts' on the basic pending registers are NOT taken into account for these two status bits. So the two pending 0,1 status bits tell you that 'there are more interrupt which you have not seen yet'.

GPU pending registers.

There are two GPU pending registers with one bit per GPU interrupt source.

7.3 Fast Interrupt (FIQ).

The ARM also supports a Fast Interrupt (FIQ). One interrupt sources can be selected to be connected to the ARM FIQ input. There is also one FIQ enable. An interrupt which is selected as FIQ should have its normal interrupt enable bit cleared. Otherwise an normal and a FIQ interrupt will be fired at the same time. Not a good idea!

7.4 Interrupt priority.

There is no priority for any interrupt. If one interrupt is much more important then all others it can be routed to the FIQ. Any remaining interrupts have to be processed by polling the pending

registers. It is up to the ARM software to device a strategy. e.g. First start looking for specific pending bits or process them all shifting one bit at a time.

As interrupt may arrive whilst this process is ongoing the usual care for any 'race-condition critical' code must be taken. The following ARM assembly code has been proven to work:

```
  .macro get_irqnr_preamble, base, tmp
  ldr  \base, =IO_ADDRESS(ARMCTRL_IC_BASE)
  .endm

  .macro get_irqnr_and_base, irqnr, irqstat, base, tmp
  ldr  \irqstat, [\base, #(ARM_IRQ_PEND0 - ARMCTRL_IC_BASE)] @ get masked status
  mov  \irqnr, #(ARM_IRQ0_BASE + 31)
  and  \tmp, \irqstat, #0x300           @ save bits 8 and 9
  bics \irqstat, \irqstat, #0x300       @ clear bits 8 and 9, and test
  bne  1010f

  tst  \tmp, #0x100
  ldrne \irqstat, [\base, #(ARM_IRQ_PEND1 - ARMCTRL_IC_BASE)]
  movne \irqnr, #(ARM_IRQ1_BASE + 31)
  @ Mask out the interrupts also present in PEND0 - see SW-5809
  bicne \irqstat, #((1<<7) | (1<<9) | (1<<10))
  bicne \irqstat, #((1<<18) | (1<<19))
  bne  1010f

  tst  \tmp, #0x200
  ldrne \irqstat, [\base, #(ARM_IRQ_PEND2 - ARMCTRL_IC_BASE)]
  movne \irqnr, #(ARM_IRQ2_BASE + 31)
  @ Mask out the interrupts also present in PEND0 - see SW-5809
  bicne \irqstat, #((1<<21) | (1<<22) | (1<<23) | (1<<24) | (1<<25))
  bicne \irqstat, #((1<<30))
  beq 1020f

1010:
  @ For non-zero x, LSB(x) = 31 - CLZ(x^(x-1))
  @ N.B. CLZ is an ARM5 instruction.
  sub  \tmp, \irqstat, #1
  eor  \irqstat, \irqstat, \tmp
  clz  \tmp, \irqstat
  sub  \irqnr, \tmp

1020: @ EQ will be set if no irqs pending
```

7.5 Registers

The base address for the ARM interrupt register is 0x7E00B000.

Registers overview:

Address offset[7]	Name	Notes
0x200	IRQ basic pending	
0x204	IRQ pending 1	
0x208	IRQ pending 2	
0x20C	FIQ control	
0x210	Enable IRQs 1	
0x214	Enable IRQs 2	
0x218	Enable Basic IRQs	
0x21C	Disable IRQs 1	
0x220	Disable IRQs 2	
0x224	Disable Basic IRQs	

The following is a table which lists all interrupts which can come from the peripherals which can be handled by the ARM.

[7] This is the offset which needs to be added to the base address to get the full hardware address.

ARM peripherals interrupts table.

#	IRQ 0-15	#	IRQ 16-31	#	IRQ 32-47	#	IRQ 48-63
0		16		32		48	smi
1		17		33		49	gpio_int[0]
2		18		34		50	gpio_int[1]
3		19		35		51	gpio_int[2]
4		20		36		52	gpio_int[3]
5		21		37		53	i2c_int
6		22		38		54	spi_int
7		23		39		55	pcm_int
8		24		40		56	
9		25		41		57	uart_int
10		26		42		58	
11		27		43	i2c_spi_slv_int	59	
12		28		44		60	
13		29	Aux int	45	pwa0	61	
14		30		46	pwa1	62	
15		31		47		63	

The table above has many empty entries. These should not be enabled as they will interfere with the GPU operation.

ARM peripherals interrupts table.

0	ARM Timer
1	ARM Mailbox
2	ARM Doorbell 0
3	ARM Doorbell 1
4	GPU0 halted (Or GPU1 halted if bit 10 of control register 1 is set)
5	GPU1 halted
6	Illegal access type 1
7	Illegal access type 0

Basic pending register.

The basic pending register shows which interrupt are pending. To speed up interrupts processing, a number of 'normal' interrupt status bits have been added to this register. This makes the *'IRQ pending base'* register different from the other *'base'* interrupt registers

Name: IRQ pend base		Address: 0x200	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:21	-	<unused>	
20	R	GPU IRQ 62	

Name: IRQ pend base		Address: 0x200	Reset: 0x000
19	R	GPU IRQ 57	
18	R	GPU IRQ 56	
17	R	GPU IRQ 55	
16	R	GPU IRQ 54	
15	R	GPU IRQ 53	
14	R	GPU IRQ 19	
13	R	GPU IRQ 18	
12	R	GPU IRQ 10	
11	R	GPU IRQ 9	
10	R	GPU IRQ 7	
9	R	One or more bits set in pending register 2	
8	R	One or more bits set in pending register 1	
7	R	Illegal access type 0 IRQ pending	
6	R	Illegal access type 1 IRQ pending	
5	R	GPU1 halted IRQ pending	
4	R	GPU0 halted IRQ pending (Or GPU1 halted if bit 10 of control register 1 is set)	
3	R	ARM Doorbell 1 IRQ pending	
2	R	ARM Doorbell 0 IRQ pending	
1	R	ARM Mailbox IRQ pending	
0	R	ARM Timer IRQ pending	

GPU IRQ x (10,11..20)

These bits are direct interrupts from the GPU. They have been selected as interrupts which are most likely to be useful to the ARM. The GPU interrupt selected are 7, 9, 10, 18, 19, 53,54,55,56,57,62. For details see the *GPU interrupts table*.

Bits set in pending registers (8,9)

These bits indicates if there are bits set in the pending 1/2 registers. The pending 1/2 registers hold ALL interrupts 0..63 from the GPU side. Some of these 64 interrupts are also connected to the basic pending register. Any bit set in pending register 1/2 which is NOT connected to the basic pending register causes bit 8 or 9 to set. Status bits 8 and 9 should be seen as "There are some interrupts pending which you don't know about. They are in pending register 1 /2."

Illegal access type-0 IRQ (7)

This bit indicate that the address/access error line from the ARM processor has generated an interrupt. That signal is asserted when either an address bit 31 or 30 was high or when an access was

seen on the ARM Peripheral bus. The status of that signal can be read from Error/HALT status register bit 2.

Illegal access type-1 IRQ (6)

This bit indicates that an address/access error is seen in the ARM control has generated an interrupt. That can either be an address bit 29..26 was high or when a burst access was seen on the GPU Peripheral bus. The status of that signal can be read from Error/HALT status register bits 0 and 1.

GPU-1 halted IRQ (5)

This bit indicate that the GPU-1 halted status bit has generated an interrupt. The status of that signal can be read from Error/HALT status register bits 4.

GPU-0 (or any GPU) halted IRQ (4)

This bit indicate that the GPU-0 halted status bit has generated an interrupt. The status of that signal can be read from Error/HALT status register bits 3.

In order to allow a fast interrupt (FIQ) routine to cope with GPU 0 OR GPU-1 there is a bit in control register 1 which, if set will also route a GPU-1 halted status on this bit.

Standard peripheral IRQs (0,1,2,3)

These bits indicate if an interrupt is pending for one of the ARM control peripherals.

GPU pending 1 register.

Name: IRQ pend base		Address: 0x204	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:0	R	IRQ pending source 31:0 (See IRQ table above)	

This register holds ALL interrupts 0..31 from the GPU side. Some of these interrupts are also connected to the basic pending register. Any interrupt status bit in here which is NOT connected to the basic pending will also cause bit 8 of the basic pending register to be set. That is all bits except 7, 9, 10, 18, 19.

GPU pending 2 register.

Name: IRQ pend base		Address: 0x208	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:0	R	IRQ pending source 63:32 (See IRQ table above)	

This register holds ALL interrupts 32..63 from the GPU side. Some of these interrupts are also connected to the basic pending register. Any interrupt status bit in here which is NOT connected to the basic pending will also cause bit 9 of the basic pending register to be set. That is all bits except . register bits 21..25, 30 (Interrupts 53..57,62).

FIQ register.

The FIQ register control which interrupt source can generate a FIQ to the ARM. Only a single interrupt can be selected.

Name: FIQ		Address: 0x20C	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:8	R	<unused>	
7	R	FIQ enable. Set this bit to 1 to enable FIQ generation. If set to 0 bits 6:0 are don't care.	
6:0	R/W	Select FIQ Source	

FIQ Source.

The FIQ source values 0-63 correspond to the GPU interrupt table. (See above)

The following values can be used to route ARM specific interrupts to the FIQ vector/routine:

FIQ index	Source
0-63	GPU Interrupts (See GPU IRQ table)
64	ARM Timer interrupt
65	ARM Mailbox interrupt
66	ARM Doorbell 0 interrupt
67	ARM Doorbell 1 interrupt
68	GPU0 Halted interrupt (Or GPU1)
69	GPU1 Halted interrupt
70	Illegal access type-1 interrupt
71	Illegal access type-0 interrupt
72-127	Do Not Use

Interrupt enable register 1.

Name: IRQ enable 1		Address: 0x210	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:0	R/Wbs	Set to enable IRQ source 31:0 (See IRQ table above)	

Writing a 1 to a bit will set the corresponding IRQ enable bit. All other IRQ enable bits are unaffected. Only bits which are enabled can be seen in the interrupt pending registers. There is no provision here to see if there are interrupts which are pending but not enabled.

Interrupt enable register 2.

Name: IRQ enable 2		Address: 0x214	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:0	R/Wbs	Set to enable IRQ source 63:32 (See IRQ table above)	

Writing a 1 to a bit will set the corresponding IRQ enable bit. All other IRQ enable bits are unaffected. Only bits which are enabled can be seen in the interrupt pending registers. There is no provision here to see if there are interrupts which are pending but not enabled.

Base Interrupt enable register.

Name: IRQ enable 3		Address: 0x218	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:8	R/Wbs	<Unused>	
7	R/Wbs	Set to enable Access error type -0 IRQ	
6	R/Wbs	Set to enable Access error type -1 IRQ	
5	R/Wbs	Set to enable GPU 1 Halted IRQ	
4	R/Wbs	Set to enable GPU 0 Halted IRQ	
3	R/Wbs	Set to enable ARM Doorbell 1 IRQ	
2	R/Wbs	Set to enable ARM Doorbell 0 IRQ	
1	R/Wbs	Set to enable ARM Mailbox IRQ	
0	R/Wbs	Set to enable ARM Timer IRQ	

Writing a 1 to a bit will set the corresponding IRQ enable bit. All other IRQ enable bits are unaffected. Again only bits which are enabled can be seen in the basic pending register. There is no provision here to see if there are interrupts which are pending but not enabled.

Interrupt disable register 1.

Name: IRQ disable 1		Address: 0x21C	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:0	R/Wbc	Set to disable IRQ source 31:0 (See IRQ table above)	

Writing a 1 to a bit will clear the corresponding IRQ enable bit. All other IRQ enable bits are unaffected.

Interrupt disable register 2.

Name: IRQ disable 2		Address: 0x220	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:0	R/Wbc	Set to disable IRQ source 63:32 (See IRQ table above)	

Writing a 1 to a bit will clear the corresponding IRQ enable bit. All other IRQ enable bits are unaffected.

Base disable register.

Name: IRQ disable 3		Address: 0x224	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:8	-	<Unused>	
7	R/Wbc	Set to disable Access error type -0 IRQ	
6	R/Wbc	Set to disable Access error type -1 IRQ	
5	R/Wbc	Set to disable GPU 1 Halted IRQ	
4	R/Wbc	Set to disable GPU 0 Halted IRQ	
3	R/Wbc	Set to disable ARM Doorbell 1 IRQ	
2	R/Wbc	Set to disable ARM Doorbell 0 IRQ	
1	R/Wbc	Set to disable ARM Mailbox IRQ	
0	R/Wbc	Set to disable ARM Timer IRQ	

Writing a 1 to a bit will clear the corresponding IRQ enable bit. All other IRQ enable bits are unaffected.

8 PCM / I2S Audio

The PCM audio interface is an APB peripheral providing input and output of telephony or high quality serial audio streams. It supports many classic PCM formats including I2S.

The PCM audio interface has 4 interface signals;

PCM_CLK - bit clock.

PCM_FS - frame sync signal.

PCM_DIN - serial data input.

PCM_DOUT - serial data output.

PCM is a serial format with a single bit data_in and single bit data_out. Data is always serialised MS-bit first.

The frame sync signal (PCM_FS) is used to delimit the serial data into individual frames. The length of the frame and the size and position of the frame sync are fully programmable.

Frames can contain 1 or 2 audio/data channels in each direction. Each channel can be between 8 and 32 bits wide and can be positioned anywhere within the frame as long as the two channels don't overlap. The channel format is separately programmable for transmit and receive directions.

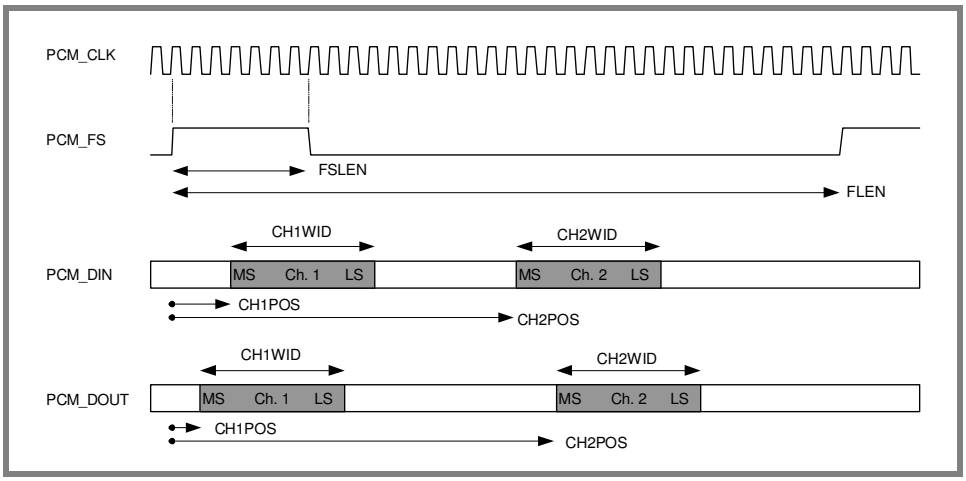


Figure 8-1 PCM Audio Interface Typical Timing

The PCM_CLK can be asynchronous to the bus APB clock and can be logically inverted if required.

The direction of the PCM_CLK and PCM_FS signals can be individually selected, allowing the interface to act as a master or slave device.

The input interface is also capable of supporting up to 2 PDM microphones, as an alternative to the classic PCM input format, in conjunction with a PCM output.

8.1 Block Diagram

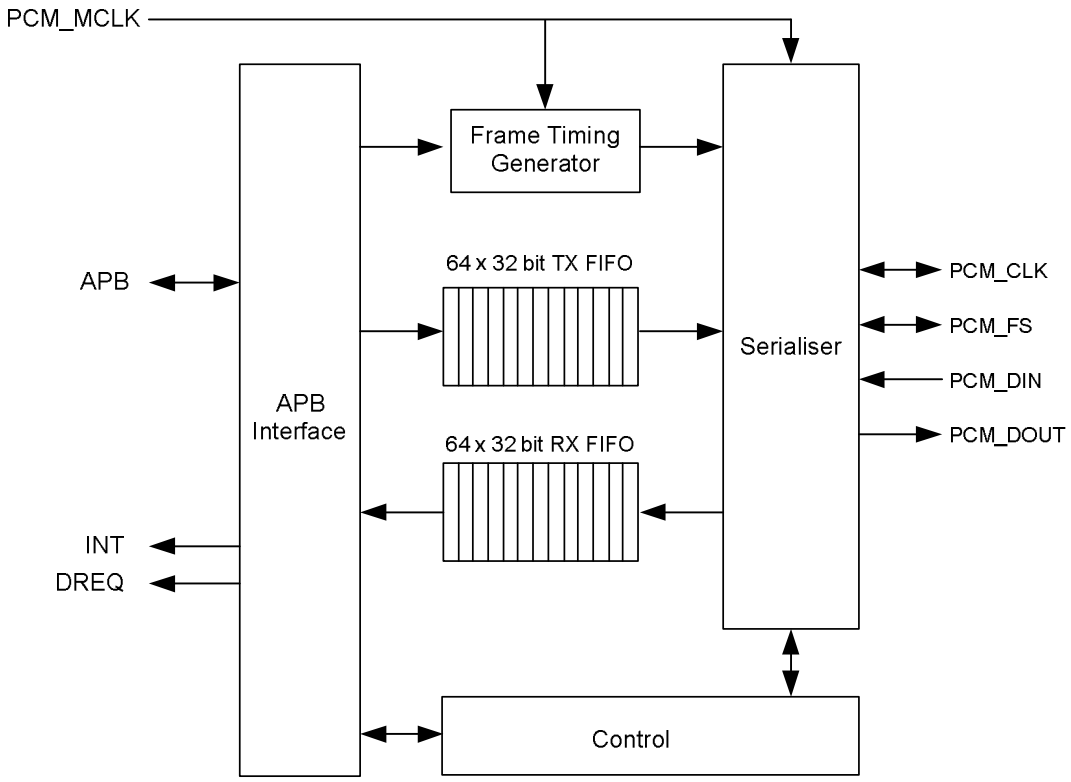


Figure 8-2 PCM Audio Interface Block Diagram

The PCM audio interface contains separate transmit and receive FIFOs. Note that if the frame contains two data channels, they must share the same FIFO and so the channel data will be interleaved. The block can be driven using simple polling, an interrupt based method or direct DMA control.

8.2 Typical Timing

Figure 8-1 shows typical interface timing and indicates the flexibility that the peripheral offers.

Normally PCM output signals change on the rising edge of PCM_CLK and input signals are sampled on its falling edge. The frame sync is considered as a data signal and sampled in the same way.

The front end of the PCM audio interface is run off the PCM_CLK and the PCM signals are timed against this clock. However, the polarity of the PCM_CLK can be physically inverted, in which case the edges are reversed.

In clock master mode (CLKM=0), the PCM_CLK is an output and is driven from the PCM_MCLK clock input.

In clock slave mode (CLKM=1), the PCM_CLK is an input, supplied by some external clock source.

In frame sync master mode (FSM=0), the PCM_FS is internally generated and is treated as a data output that changes on the positive edge of the clock. The length and polarity of the frame sync is fully programmable and it can be used as a standard frame sync signal, or as an L-R signal for I2S.

In frame sync slave mode (FSM=1), the PCM_FS is treated as a data input and is sampled on the negative edge of PCM_CLK. The first clock of a frame is taken as the first clock period where PCM_FS is sampled as a 1 following a period or periods where it was previously a 0. The PCM audio interface locks onto the incoming frame sync and uses this to indicate where the data channels are positioned. The precise timing at the start of frame is shown in Figure 8-3.

Note that in frame sync slave mode there are two synchronising methods. The legacy method is used when the frame length = 0. In this case the internal frame logic has to detect the incoming PCM_FS signal and reset the internal frame counter at the start of every frame. The logic relies on the PCM_FS to indicate the length of the frame and so can cope with adjacent frames of different lengths. However, this creates a short timing path that will corrupt the PCM_DOUT for one specific frame/channel setting.

The preferred method is to set the frame length to the expected length. Here the incoming PCM_FS is used to resynchronise the internal frame counter and this eliminates the short timing path.

8.3 Operation

The PCM interface runs asynchronously at the PCM_CLK rate and automatically transfers transmit and receive data across to the internal APB clock domain. The control registers are NOT synchronised and should be programmed before the device is enabled and should NOT be changed whilst the interface is running.

Only the EN, RXON and TXON bits of the PCMCS register are synchronised across the PCM - APB clock domain and are allowed to be changed whilst the interface is running.

The EN bit is a global power-saving enable. The TXON and RXON bits enable transmit and receive, and the interface is running whenever either TXON or RXON is enabled.

In operation, the PCM format is programmed by setting the appropriate frame length, frame sync, channel position values, and signal polarity controls. The transmit FIFO should be preloaded with data and the interface can then be enabled and started, and will run continuously until stopped. If the transmit FIFO becomes empty or the receive FIFO becomes full, the RXERR or TXERR error flags will be set, but the interface will just continue. If the RX FIFO overflows, new samples are discarded and if the TX FIFO underflows, zeros are transmitted.

Normally channel data is read or written into the appropriate FIFO as a single word. If the channel is less than 32 bits, the data is right justified and should be padded with zeros. If the RXSEX bit is set then the received data is sign extended up to the full 32 bits. When a frame is programmed to have two data channels, then each channel is written/read as a separate word in the FIFO, producing an interleaved data stream. When initialising the interface, the first word read out of the TX FIFO will be used for the first channel, and the data from the first channel on the first frame to be received will be the first word written into the RX FIFO.

If a FIFO error occurs in a two channel frame, then channel synchronisation may be lost which may result in a left right audio channel swap. RXSYNC and TXSYNC status bits are

provided to help determine if channel slip has occurred. They indicate if the number of words in the FIFO is a multiple of a full frame (taking into account where we are in the current frame being transferred). This assumes that an integer number of frames data has been sent/read from the FIFOs.

If a frame is programmed to have two data channels and the packed mode bits are set (FRXP FTXP) then the FIFOs are configured so that each word contains the data for both channels (2x 16 bit samples). In this mode each word written to the TX FIFO contains 2 16 bit samples, and the Least Significant sample is transmitted first. Each word read from the RX FIFO will contain the data received from 2 channels, the first channel received will be in the Least Significant half of the word. If the channels size is less than 16 bits, the TX data will be truncated and RX data will be padded to 16 bits with zeros.

Note that data is always serialised MS-bit first. This is well-established behaviour in both PCM and I2S.

If the PDM input mode is enabled then channel 1 is sampled on the negative edge of PCM_CLK whilst channel 2 is sampled on the positive edge of PCM_CLK.

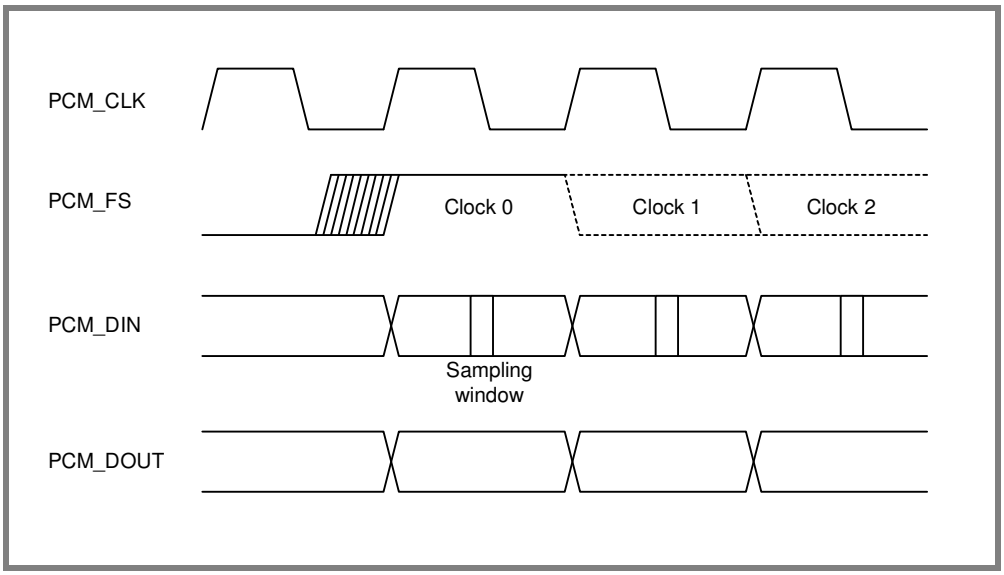


Figure 8-3 Timing at Start of Frame

Note that the precise timing of FS (when it is an input) is not clearly defined and it may change state before or after the positive edge of the clock. Here the first clock of the frame is defined as the clock period where the PCM_FS is sampled (negative edge) as a 1 where it was previously sampled as a 0.

8.4 Software Operation

8.4.1 Operating in Polled mode

Set the EN bit to enable the PCM block. Set all operational values to define the frame and channel settings. Assert RXCLR and/or TXCLR wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed. Set RXTHR/TXTHR to determine the FIFO thresholds.

If transmitting, ensure that sufficient sample words have been written to PCMFIFO before transmission is started. Set TXON and/or RXON to begin operation. Poll TXW writing sample words to PCMFIFO and RXR reading sample words from PCMFIFO until all data is transferred.

8.4.2 Operating in Interrupt mode

a) Set the EN bit to enable the PCM block. Set all operational values to define the frame and channel settings. Assert RXCLR and/or TXCLR wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed. Set RXTHR/TXTHR to determine the FIFO thresholds.

b) Set INTR and/or INTT to enable interrupts.

c) If transmitting, ensure that sufficient sample words have been written to PCMFIFO before transmission is started. Set TXON and/or RXON to begin operation.

d) When an interrupt occurs, check RXR. If this is set then one or more sample words are available in PCMFIFO. If TXW is set then one or more sample words can be sent to PCMFIFO.

8.4.3 DMA

a) Set the EN bit to enable the PCM block. Set all operational values to define the frame and channel settings. Assert RXCLR and/or TXCLR wait for 2 PCM clocks to ensure the FIFOs are reset. The SYNC bit can be used to determine when 2 clocks have passed.

b) Set DMAEN to enable DMA DREQ generation and set RXREQ/TXREQ to determine the FIFO thresholds for the DREQs. If required, set TXPANIC and RXPANIC to determine the level at which the DMA should increase its AXI priority,

c) In the DMA controllers set the correct DREQ channels, one for RX and one for TX. Start the DMA which should fill the TX FIFO.

d) Set TXON and/or RXON to begin operation.

8.5 Error Handling.

In all software operational modes, the possibility of FIFO over or under run exists. Should this happen when using 2 channels per frame, there is a risk of losing sync with the channel data stored in the FIFO. If this happens and is not detected and corrected, then the data channels may become swapped.

The FIFO's will automatically detect an error condition caused by a FIFO over or under-run and this will set the appropriate latching error bit in the control/status register. Writing a '1' back to this error bit will clear the latched flag.

In a system using a polled operation, the error bits can be checked manually. For an interrupt or DMA based system, setting the INTE bit will cause the PCM interface to generate an interrupt when an error is detected.

If a FIFO error occurs during operation in which 2 data channels are being used then the synchronisation of the data may be lost. This can be recovered by either of these two methods:

a) Disable transmit and receive (TXON and RXON =0). Clear the FIFO's (RXCLR and TXCLR =1). Note that it may take up to 2 PCM clocks for the FIFOs to be physically cleared after initiating a clear. Then preload the transmit FIFO and restart transmission. This of course loses the data in the FIFO and further interrupts the data flow to the external device.

b) Examine the TXSYNC and RXSYNC flags. These flags indicate if the amount of data in the FIFO is a whole number of frames, automatically taking into account where we are in the current frame being transmitted or received. Thus, providing an even number of samples was read or written to the FIFOs, then if the flags are set then this indicates that a single word needs to be written or read to adjust the data. Normal exchange of data can then proceed (where the first word in a data pair is for channel 1). This method should cause less disruption to the data stream.

8.6 PDM Input Mode Operation

The PDM input mode is capable of interfacing with two digital half-cycle PDM microphones and implements a 4th order CIC decimation filter with a selectable decimation factor. The clock input of the microphones is shared with the PCM output codec and it should be configured to provide the correct clock rate for the microphones. As a result it may be necessary to add a number of padding bits into the PCM output and configure the output codec to allow for this.

When using the PDM input mode the bit width and the rate of the data received will depend on the decimation factor used. Once the data has been read from the peripheral a further decimation and filtering stage will be required and can be implemented in software. The software filter should also correct the droop introduced by the CIC filter stage. Similarly a DC correction stage should also be employed.

PDMN	PCM_CLK (MHz)	Peripheral Output Format	OSR	Fs
0 (N=16)	3.072	16 bits unsigned	4	48kHz
1 (N=32)	3.072	20 bits unsigned	2	48kHz

Table 8-1 PDM Input Mode Configuration

8.7 GRAY Code Input Mode Operation

GRAY mode is used for an incoming data stream only. GRAY mode is selected by setting the enable bit (EN) in the PCM_GRAY register.

In this mode data is received on the PCM_DIN (data) and the PCM_FS (strobe) pins. The data is expected to be in data/strobe format. In this mode data is detected when either the data or the strobe change state. As each bit is received it is written into the RX buffer and when 32 bits are received they are written out to the RXFIFO as a 32 bit word. In order for this mode to work the user must program a PCM clock rate which is 4 times faster then the gray data rate. Also the gray coded data input signals should be clean.

The normal RXREQ and RXTHR FIFO levels will apply as for normal PCM received data.

If a message is received that is not a multiple of 32 bits, any data in the RX Buffer can be flushed out by setting the flush bit (FLUSH). Once set, this bit will read back as zero until the flush operation has completed. This may take several cycles as the APB clock may be many times faster than the PCM clock. Once the flush has occurred, the bits are packed up to 32 bits with zeros and written out to the RXFIFO. The flushed field (FLUSHED) will indicate how many of bits of this word are valid.

Note that to get an accurate indication of the number of bits currently in the rx shift register (RXLEVEL) the APB clock must be at least 2x the PCM_CLK.

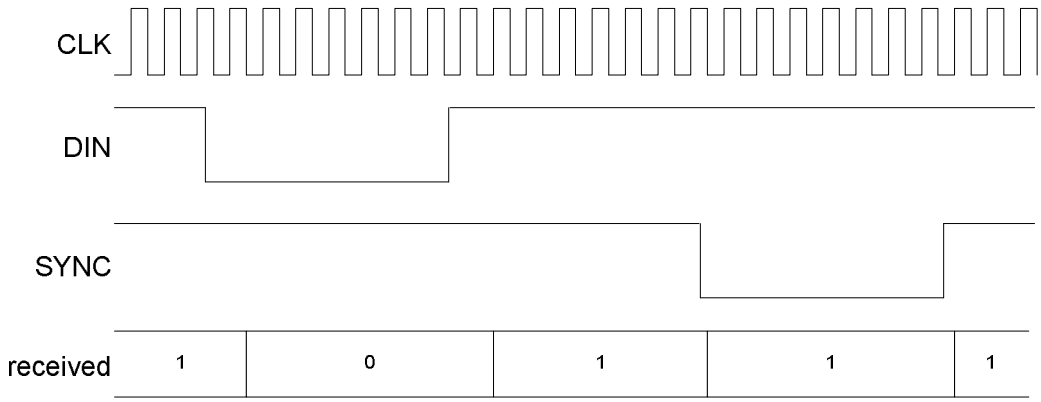


Figure 8-4 Gray mode input format

8.8 PCM Register Map

There is only PCM module in the BCM2835. The PCM base address for the registers is 0x7E203000.

PCM Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	CS_A	PCM Control and Status	32
0x4	FIFO_A	PCM FIFO Data	32
0x8	MODE_A	PCM Mode	32
0xc	RXC_A	PCM Receive Configuration	32
0x10	TXC_A	PCM Transmit Configuration	32
0x14	DREQ_A	PCM DMA Request Level	32

0x18	INTEN_A	PCM Interrupt Enables	32
0x1c	INTSTC_A	PCM Interrupt Status & Clear	32
0x20	GRAY	PCM Gray Mode Control	32

CS_A Register

Synopsis This register contains the main control and status bits for the PCM. The bottom 3 bits of this register can be written to whilst the PCM is running. The remaining bits cannot.

Bit(s)	Field Name	Description	Type	Reset
31:26		***Reserved*** *- Write as 0, read as don't care*		
25	STBY	RAM Standby This bit is used to control the PCM Rams standby mode. By default this bit is 0 causing RAMs to start initially in standby mode. Rams should be released from standby prior to any transmit/receive operation. Allow for at least 4 PCM clock cycles to take effect. This may or may not be implemented, depending upon the RAM libraries being used.	RW	0x0
24	SYNC	PCM Clock sync helper. This bit provides a software synchronisation mechanism to allow the software to detect when 2 PCM clocks have occurred. It takes 2 PCM clocks before the value written to this bit will be echoed back in the read value.	RW	0x0
23	RXSEX	RX Sign Extend 0 = No sign extension. 1 = Sign extend the RX data. When set, the MSB of the received data channel (as set by the CHxWID parameter) is repeated in all the higher data bits up to the full 32 bit data width.	RW	0x0
22	RXF	RX FIFO is Full 0 = RX FIFO can accept more data. 1 = RX FIFO is full and will overflow if more data is received.	RO	0x0

21	TXE	TX FIFO is Empty 0 = TX FIFO is not empty. 1 = TX FIFO is empty and underflow will take place if no more data is written.	RO	0x1
20	RXD	Indicates that the RX FIFO contains data 0 = RX FIFO is empty. 1 = RX FIFO contains at least 1 sample.	RO	0x0
19	TXD	Indicates that the TX FIFO can accept data 0 = TX FIFO is full and so cannot accept more data. 1 = TX FIFO has space for at least 1 sample.	RO	0x1
18	RXR	Indicates that the RX FIFO needs reading 0 = RX FIFO is less than RXTHR full. 1 = RX FIFO is RXTHR or more full. This is cleared by reading sufficient data from the RX FIFO.	RO	0x0
17	TXW	Indicates that the TX FIFO needs Writing 0 = TX FIFO is at least TXTHR full. 1 = TX FIFO is less then TXTHR full. This is cleared by writing sufficient data to the TX FIFO.	RO	0x1
16	RXERR	RX FIFO Error 0 = FIFO has had no errors. 1 = FIFO has had an under or overflow error. This flag is cleared by writing a 1.	RW	0x0
15	TXERR	TX FIFO Error 0 = FIFO has had no errors. 1 = FIFO has had an under or overflow error. This flag is cleared by writing a 1.	RW	0x0
14	RXSYNC	RX FIFO Sync 0 = FIFO is out of sync. The amount of data left in the FIFO is not a multiple of that required for a frame. This takes into account if we are halfway through the frame. 1 = FIFO is in sync.	RO	0x0
13	TXSYNC	TX FIFO Sync 0 = FIFO is out of sync. The amount of data left in the FIFO is not a multiple of that required for a frame. This takes into account if we are halfway through the frame. 1 = FIFO is in sync.	RO	0x0
12:10		***Reserved*** *- Write as 0, read as don't care*		

9	DMAEN	DMA DREQ Enable 0 = Don t generate DMA DREQ requests. 1 = Generates a TX DMA DREQ requests whenever the TX FIFO level is lower than TXREQ or generates a RX DMA DREQ when the RX FIFO level is higher than RXREQ.	RW	0x0
8:7	RXTHR	Sets the RX FIFO threshold at which point the RXR flag is set 00 = set when we have a single sample in the RX FIFO 01 = set when the RX FIFO is at least full 10 = set when the RX FIFO is at least 11 = set when the RX FIFO is full	RW	0x0
6:5	TXTHR	Sets the TX FIFO threshold at which point the TXW flag is set 00 = set when the TX FIFO is empty 01 = set when the TX FIFO is less than full 10 = set when the TX FIFO is less than full 11 = set when the TX FIFO is full but for one sample	RW	0x0
4	RXCLR	Clear the RX FIFO . Assert to clear RX FIFO. This bit is self clearing and is always read as clear Note that it will take 2 PCM clocks for the FIFO to be physically cleared.	WO	0x0
3	TXCLR	Clear the TX FIFO Assert to clear TX FIFO. This bit is self clearing and is always read as clear. Note that it will take 2 PCM clocks for the FIFO to be physically cleared.	WO	0x0
2	TXON	Enable transmission 0 = Stop transmission. This will stop immediately if possible or else at the end of the next frame. The TX FIFO can still be written to to preload data. 1 = Start transmission. This will start transmitting at the start of the next frame. Once enabled, the first data read from the TX FIFO will be placed in the first channel of the frame, thus ensuring proper channel synchronisation. The frame counter will be started whenever TXON or RXON are set. This bit can be written whilst the interface is running.	RW	0x0

1	RXON	Enable reception. 0 = Disable reception. This will stop on the next available frame end. RX FIFO data can still be read. 1 = Enable reception. This will be start receiving at the start of the next frame. The first channel to be received will be the first word written to the RX FIFO. This bit can be written whilst the interface is running.	RW	0x0
0	EN	Enable the PCM Audio Interface 0 = The PCM interface is disabled and most logic is gated off to save power. 1 = The PCM Interface is enabled. This bit can be written whilst the interface is running.	RW	0x0

FIFO_A Register

Synopsis This is the FIFO port of the PCM. Data written here is transmitted, and received data is read from here.

Bit(s)	Field Name	Description	Type	Reset
31:0		***Reserved*** *- Write as 0, read as don't care*		

MODE_A Register

Synopsis This register defines the basic PCM Operating Mode. It is used to configure the frame size and format and whether the PCM is in master or slave modes for its frame sync or clock. This register cannot be changed whilst the PCM is running.

Bit(s)	Field Name	Description	Type	Reset
31:29		***Reserved*** *- Write as 0, read as don't care*		

28	CLK_DIS	PCM Clock Disable 1 = Disable the PCM Clock. This cleanly disables the PCM clock. This enables glitch free clock switching between an internal and an uncontrollable external clock. The PCM clock can be disabled, and then the clock source switched, and then the clock re-enabled. 0 = Enable the PCM clock.	RW	0x0
27	PDMN	PDM Decimation Factor (N) 0 = Decimation factor 16. 1 = Decimation factor 32. Sets the decimation factor of the CIC decimation filter.	RW	0x0
26	PDME	PDM Input Mode Enable 0 = Disable PDM (classic PCM input). 1 = Enable PDM input filter. Enable CIC filter on input pin for PDM inputs. In order to receive data RXON must also be set.	RW	0x0
25	FRXP	Receive Frame Packed Mode 0 = The data from each channel is written into the RX FIFO. 1 = The data from both RX channels is merged (1st channel is in the LS half) and then written to the RX FIFO as a single 2x16 bit packed mode word. First received channel in the frame goes into the LS half word. If the received data is larger than 16 bits, the upper bits are truncated. The maximum channel size is 16 bits.	RW	0x0
24	FTXP	Transmit Frame Packed Mode 0 = Each TX FIFO word is written into a single channel. 1 = Each TX FIFO word is split into 2 16 bit words and used to fill both data channels in the same frame. The maximum channel size is 16 bits. The LS half of the word is used in the first channel of the frame.	RW	0x0
23	CLKM	PCM Clock Mode 0 = Master mode. The PCM CLK is an output and drives at the MCLK rate. 1 = Slave mode. The PCM CLK is an input.	RW	0x0

22	CLKI	Clock Invert this logically inverts the PCM_CLK signal. 0 = Outputs change on rising edge of clock, inputs are sampled on falling edge. 1 = Outputs change on falling edge of clock, inputs are sampled on rising edge.	RW	0x0
21	FSM	Frame Sync Mode 0 = Master mode. The PCM_FS is an output and we generate the frame sync. 1 = Slave mode. The PCM_FS is an input and we lock onto the incoming frame sync signal.	RW	0x0
20	FSI	Frame Sync Invert This logically inverts the frame sync signal. 0 = In master mode, FS is normally low and goes high to indicate frame sync. In slave mode, the frame starts with the clock where FS is a 1 after being a 0. 1 = In master mode, FS is normally high and goes low to indicate frame sync. In slave mode, the frame starts with the clock where FS is a 0 after being a 1.	RW	0x0
19:10	FLEN	Frame Length Sets the frame length to (FLEN+1) clocks. Used only when FSM == 0. 1 = frame length of 2 clocks. 2 = frame length of 3 clocks. etc	RW	0x0
9:0	FSLEN	Frame Sync Length Sets the frame sync length to (FSLEN) clocks. This is only used when FSM == 0. PCM_FS will remain permanently active if FSLEN >= FLEN. 0 = frame sync pulse is off. 1 = frame sync pulse is 1 clock wide. etc	RW	0x0

RXC_A Register

Synopsis Sets the Channel configurations for Receiving. This sets the position and width of the 2 receive channels within the frame. The two channels cannot overlap, however they channel 1 can come after channel zero, although the first data will always be from the first channel in the frame. Channels can also straddle the frame begin end boundary as that is set by the frame sync position. This register cannot be changed whilst the PCM is running.

Bit(s)	Field Name	Description	Type	Reset

31	CH1WEX	Channel 1 Width Extension Bit This is the MSB of the channel 1 width (CH1WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM	RW	0x0
30	CH1EN	Channel 1 Enable 0 = Channel 1 disabled and no data is received from channel 1 and written to the RX FIFO. 1 = Channel 1 enabled.	RW	0x0
29:20	CH1POS	Channel 1 Position This sets the bit clock at which the first bit (MS bit) of channel 1 data occurs in the frame. 0 indicates the first clock of frame.	RW	0x0
19:16	CH1WID	Channel 1 Width This sets the width of channel 1 in bit clocks. This field has been extended with the CH1WEX bit giving a total width of (CH1WEX* 16) + CH1WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide	RW	0x0
15	CH2WEX	Channel 2 Width Extension Bit This is the MSB of the channel 2 width (CH2WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM	RW	0x0
14	CH2EN	Channel 2 Enable 0 = Channel 2 disabled and no data is received from channel 2 and written to the RX FIFO. 1 = Channel 2 enabled.	RW	0x0
13:4	CH2POS	Channel 2 Position This sets the bit clock at which the first bit (MS bit) of channel 2 data occurs in the frame. 0 indicates the first clock of frame.	RW	0x0
3:0	CH2WID	Channel 2 Width This sets the width of channel 2 in bit clocks. This field has been extended with the CH2WEX bit giving a total width of (CH2WEX* 16) + CH2WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide	RW	0x0

TXC_A Register

Synopsis Sets the Channel configurations for Transmitting. This sets the position and width of the 2 transmit channels within the frame. The two channels cannot overlap, however they channel 1 can come after channel zero, although the first data will always be used in the first channel in the frame. Channels can also straddle the frame begin end boundary as that is set by the frame sync position. This register cannot be changed whilst the PCM is running.

Bit(s)	Field Name	Description	Type	Reset
31	CH1WEX	Channel 1 Width Extension Bit This is the MSB of the channel 1 width (CH1WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM	RW	0x0
30	CH1EN	Channel 1 Enable 0 = Channel 1 disabled and no data is taken from the TX FIFO and transmitted on channel 1. 1 = Channel 1 enabled.	RW	0x0
29:20	CH1POS	Channel 1 Position This sets the bit clock at which the first bit (MS bit) of channel 1 data occurs in the frame. 0 indicates the first clock of frame.	RW	0x0
19:16	CH1WID	Channel 1 Width This sets the width of channel 1 in bit clocks. This field has been extended with the CH1WEX bit giving a total width of (CH1WEX* 16) + CH1WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide	RW	0x0
15	CH2WEX	Channel 2 Width Extension Bit This is the MSB of the channel 2 width (CH2WID). It allows widths greater than 24 bits to be programmed and is added here to keep backwards compatibility with older versions of the PCM	RW	0x0
14	CH2EN	Channel 2 Enable 0 = Channel 2 disabled and no data is taken from the TX FIFO and transmitted on channel 2. 1 = Channel 2 enabled.	RW	0x0

13:4	CH2POS	Channel 2 Position This sets the bit clock at which the first bit (MS bit) of channel 2 data occurs in the frame. 0 indicates the first clock of frame.	RW	0x0
3:0	CH2WID	Channel 2 Width This sets the width of channel 2 in bit clocks. This field has been extended with the CH2WEX bit giving a total width of (CH2WEX* 16) + CH2WID + 8. The Maximum supported width is 32 bits. 0 = 8 bits wide 1 = 9 bits wide	RW	0x0

DREQ_A Register

Synopsis Set the DMA DREQ and Panic thresholds. The PCM drives 2 DMA controls back to the DMA, one for the TX channel and one for the RX channel. DMA DREQ is used to request the DMA to perform another transfer, and DMA Panic is used to tell the DMA to use its panic level of priority when requesting thins on the AXI bus. This register cannot be changed whilst the PCM is running.

Bit(s)	Field Name	Description	Type	Reset
31		***Reserved*** *- Write as 0, read as don't care*		
30:24	TX_PANIC	TX Panic Level This sets the TX FIFO Panic level. When the level is below this the PCM will assert its TX DMA Panic signal.	RW	0x10
23		***Reserved*** *- Write as 0, read as don't care*		
22:16	RX_PANIC	RX Panic Level This sets the RX FIFO Panic level. When the level is above this the PCM will assert its RX DMA Panic signal.	RW	0x30
15		***Reserved*** *- Write as 0, read as don't care*		
14:8	TX	TX Request Level This sets the TX FIFO DREQ level. When the level is below this the PCM will assert its DMA DREQ signal to request more data is written to the TX FIFO.	RW	0x30
7		***Reserved*** *- Write as 0, read as don't care*		

6:0	RX	RX Request Level This sets the RX FIFO DREQ level. When the level is above this the PCM will assert its DMA DREQ signal to request that some more data is read out of the RX FIFO.	RW	0x20

INTEN_A Register

Synopsis Set the reasons for generating an Interrupt. This register cannot be changed whilst the PCM is running.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3	RXERR	RX Error Interrupt Setting this bit enables interrupts from PCM block when RX FIFO error occurs.	RW	0x0
2	TXERR	TX Error Interrupt Setting this bit enables interrupts from PCM block when TX FIFO error occurs.	RW	0x0
1	RXR	RX Read Interrupt Enable Setting this bit enables interrupts from PCM block when RX FIFO level is greater than or equal to the specified RXTHR level.	RW	0x0
0	TXW	TX Write Interrupt Enable Setting this bit enables interrupts from PCM block when TX FIFO level is less than the specified TXTHR level.	RW	0x0

INTSTC_A Register

Synopsis This register is used to read and clear the PCM interrupt status. Writing a 1 to the asserted bit clears the bit. Writing a 0 has no effect.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		

3	RXERR	RX Error Interrupt Status / Clear This bit indicates an interrupt occurred on RX FIFO Error. Writing 1 to this bit clears it. Writing 0 has no effect.	RW	0x0
2	TXERR	TX Error Interrupt Status / Clear This bit indicates an interrupt occurred on TX FIFO Error. Writing 1 to this bit clears it. Writing 0 has no effect.	RW	0x0
1	RXR	RX Read Interrupt Status / Clear This bit indicates an interrupt occurred on RX Read. Writing 1 to this bit clears it. Writing 0 has no effect.	RW	0x0
0	TXW	TX Write Interrupt Status / Clear This bit indicates an interrupt occurred on TX Write. Writing 1 to this bit clears it. Writing 0 has no effect.	RW	0x0

GRAY Register

Synopsis This register is used to control the gray mode generation. This is used to put the PCM into a special data/strobe mode. This mode is under 'best effort ' contract.

Bit(s)	Field Name	Description	Type	Reset
31:22		***Reserved*** *- Write as 0, read as don't care*		
21:16	RXFIFOLEVEL	The Current level of the RXFIFO This indicates how many words are currently in the RXFIFO.	RO	0x0
15:10	FLUSHED	The Number of bits that were flushed into the RXFIFO This indicates how many bits were valid when the flush operation was performed. The valid bits are from bit 0 upwards. Non-valid bits are set to zero.	RO	0x0
9:4	RXLEVEL	The Current fill level of the RX Buffer This indicates how many GRAY coded bits have been received. When 32 bits are received, they are written out into the RXFIFO.	RO	0x0

3		***Reserved*** *- Write as 0, read as don't care*		
2	FLUSH	Flush the RX Buffer into the RX FIFO This forces the RX Buffer to do an early write. This is necessary if we have reached the end of the message and we have bits left in the RX Buffer. Flushing will write these bits as a single 32 bit word, starting at bit zero. Empty bits will be packed with zeros. The number of bits written will be recorded in the FLUSHED Field. This bit is written as a 1 to initiate a flush. It will read back as a zero until the flush operation has completed (as the PCM Clock may be very slow).	RW	0x0
1	CLR	Clear the GRAY Mode Logic This Bit will reset all the GRAY mode logic, and flush the RX buffer. It is not self clearing.	RW	0x0
0	EN	Enable GRAY Mode Setting this bit will put the PCM into GRAY mode. In gray mode the data is received on the data in and the frame sync pins. The data is expected to be in data/strobe format.	RW	0x0

9 Pulse Width Modulator

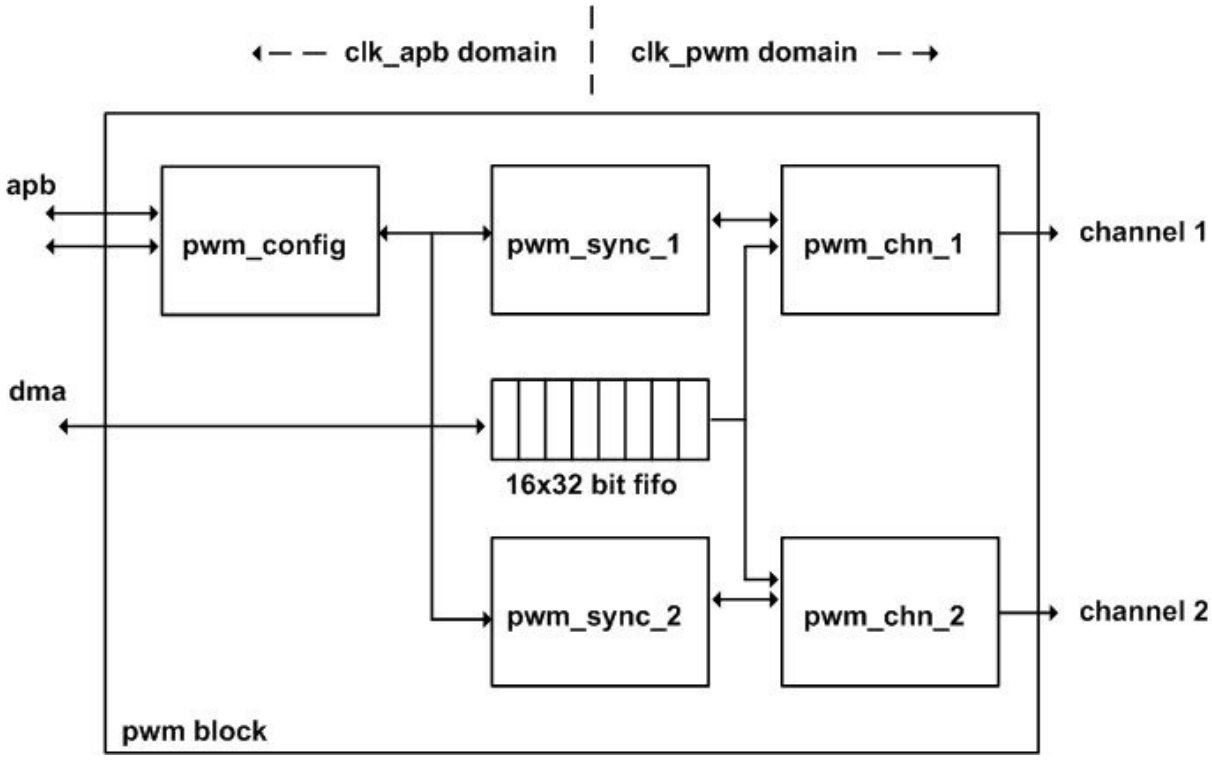
9.1 Overview

This section specifies in detail the functionality provided by the device Pulse Width Modulator (PWM) peripheral.

The PWM controller incorporates the following features:

- Two independent output bit-streams, clocked at a fixed frequency.
- Bit-streams configured individually to output either PWM or a serialised version of a 32-bit word.
- PWM outputs have variable input and output resolutions.
- Serialise mode configured to load data to and/or read data from a FIFO storage block, which can store up to eight 32-bit words.
- Both modes clocked by clk_pwm which is nominally 100MHz, but can be varied by the clock manager.

9.2 Block Diagram



9.3 PWM Implementation

A value represented as a ratio of N/M can be transmitted along a serial channel with pulse width modulation in which the value is represented by the duty cycle of the output signal. To send value N/M within a periodic sequence of M cycles, output should be 1 for N cycles and 0 for (M-N) cycles. The desired sequence should have 1's and 0's spread out as even as possible so that during any arbitrary period of time duty cycle achieves closest approximation of the value. This can be shown in the following table where 4/8 is modulated (N= 4, M= 8).

Bad	0	0	0	0	1	1	1	1	0	0	0	0
Fair	0	0	1	1	0	0	1	1	0	0	1	1
Good	0	1	0	1	0	1	0	1	0	1	0	1

Sequence which gives the 'good' approximation from the table above can be achieved by the following algorithm:

```
1. Set context = 0
2. context = context + N
3.  if (context >= M)
        context = context – M
        send 1
    else
```

where context is a register which stores the result of the addition/subtractions.

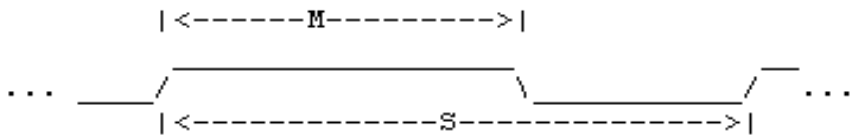
9.4 Modes of Operation

PWM controller consists of two independent channels (pwm_chn in block diagram) which implement the pwm algorithm explained in section 1.3. Each channel can operate in either pwm mode or serialiser mode.

PWM mode: There are two sub-modes in PWM mode: MSEN=0 and MSEN=1.

When MSEN=0, which is the default mode, data to be sent is interpreted as the value N of the algorithm explained above. Number of clock cycles (range) used to send data is the value M of the algorithm. Pulses are sent within this range so that the resulting duty cycle is N/M. Channel sends its output continuously as long as data register is used, or buffer is used and it is not empty.

When MSEN=1, PWM block does not use the algorithm explained above, instead it sends serial data with the M/S ratio as in the picture below. M is the data to be sent, and S is the range. This mode may be preferred if high frequency modulation is not required or has negative effects. Channel sends its output continuously as long as data register is used, or buffer is used and it is not empty.



Serial bit transmission when M/S Mode enabled

Serialiser mode: Each channel is also capable of working as a serialiser. In this mode data written in buffer or the data register is sent serially.

9.5 Quick Reference

- PWM DMA is mapped to DMA channel 5.
- GPIOs are assigned to PWM channels as below. Please refer to GPIO section for further details:

	PWM0	PWM1
GPIO 12	Alt Fun 0	-
GPIO 13	-	Alt Fun 0
GPIO 18	Alt Fun 5	-
GPIO 19	-	Alt Fun 5
GPIO 40	Alt Fun 0	-
GPIO 41	-	Alt Fun 0
GPIO 45	-	Alt Fun 0
GPIO 52	Alt Fun 1	-
GPIO 53	-	Alt Fun 1

- PWM clock source and frequency is controlled in CPRMAN.

9.6 Control and Status Registers

PWM Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	CTL	PWM Control	32
0x4	STA	PWM Status	32
0x8	DMAC	PWM DMA Configuration	32
0x10	RNG1	PWM Channel 1 Range	32
0x14	DAT1	PWM Channel 1 Data	32
0x18	FIF1	PWM FIFO Input	32
0x20	RNG2	PWM Channel 2 Range	32
0x24	DAT2	PWM Channel 2 Data	32

CTL Register

Synopsis PWENi is used to enable/disable the corresponding channel. Setting this bit to 1 enables the channel and transmitter state machine. All registers and FIFO is writable without setting this bit.
MODEi bit is used to determine mode of operation. Setting this bit to 0 enables PWM mode. In this mode data stored in either PWM_DATi or FIFO is transmitted by pulse width modulation within the range defined by PWM_RNGi. When this mode is used MSENi defines whether to use PWM algorithm. Setting MODEi to 1 enables serial mode, in which data stored in either PWM_DATi or FIFO is transmitted serially within the range defined by PWM_RNGi. Data is transmitted MSB first and truncated or zero-padded depending on PWM_RNGi. Default mode is PWM.
RPTLi is used to enable/disable repeating of the last data available in the FIFO just before it empties. When this bit is 1 and FIFO is used, the last available data in the FIFO is repeatedly sent. This may be useful in PWM mode to avoid duty cycle gaps. If the FIFO is not used this bit does not have any effect. Default operation is do-not-repeat.
SBITi defines the state of the output when no transmission takes place. It also defines the zero polarity for the zero padding in serialiser mode. This bit is padded between two consecutive transfers as well as tail of the data when PWM_RNGi is larger than bit depth of data being transferred. this bit is zero by default.
POLAi is used to configure the polarity of the output bit. When set to high the final output is inverted. Default operation is no inversion.
USEFi bit is used to enable/disable FIFO transfer. When this bit is high data stored in the FIFO is used for transmission. When it is low, data written to PWM_DATi is transferred. This bit is 0 as default.
CLRF is used to clear the FIFO. Writing a 1 to this bit clears the FIFO. Writing 0 has no effect. This is a single shot operation and reading the bit always returns 0.
MSENi is used to determine whether to use PWM algorithm or simple M/S ratio transmission. When this bit is high M/S transmission is used. This bit is zero as default. When MODEi is 1, this configuration bit has no effect.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		
15	MSEN2	Channel 2 M/S Enable 0: PWM algorithm is used 1: M/S transmission is used.	RW	0x0
14		***Reserved*** *- Write as 0, read as don't care*		
13	USEF2	Channel 1 Use Fifo 0: Data register is transmitted 1: Fifo is used for transmission	RW	0x0
12	POLA2	Channel 1 Polarity 0 : 0=low 1=high 1: 1=low 0=high	RW	0x0
11	SBIT2	Channel 1 Silence Bit Defines the state of the output when no transmission takes place	RW	0x0

10	RPTL2	Channel 1 Repeat Last Data 0: Transmission interrupts when FIFO is empty 1: Last data in FIFO is transmitted repetedly until FIFO is not empty	RW	0x0
9	MODE2	Channel 1 Mode 0: PWM mode 1: Serialiser mode	RW	0x0
8	PWEN2	Channel 1 Enable 0: Channel is disabled 1: Channel is enabled	RW	0x0
7	MSEN1	Channel 1 M/S Enable 0: PWM algorithm is used 1: M/S transmission is used.	RW	0x0
6	CLRF1	Clear Fifo 1: Clears FIFO 0: Has no effect This is a single shot operation. This bit always reads 0	RO	0x0
5	USEF1	Channel 1 Use Fifo 0: Data register is transmitted 1: Fifo is used for transmission	RW	0x0
4	POLA1	Channel 1 Polarity 0 : 0=low 1=high 1: 1=low 0=high	RW	0x0
3	SBIT1	Channel 1 Silence Bit Defines the state of the output when no transmission takes place	RW	0x0
2	RPTL1	Channel 1 Repeat Last Data 0: Transmission interrupts when FIFO is empty 1: Last data in FIFO is transmitted repetedly until FIFO is not empty	RW	0x0
1	MODE1	Channel 1 Mode 0: PWM mode 1: Serialiser mode	RW	0x0
0	PWEN1	Channel 1 Enable 0: Channel is disabled 1: Channel is enabled	RW	0x0

STA Register

Synopsis FULL1 bit indicates the full status of the FIFO. If this bit is high FIFO is full.
EMPT1 bit indicates the empty status of the FIFO. If this bit is high FIFO is empty.
WERR1 bit sets to high when a write when full error occurs. Software must clear this bit by writing 1. Writing 0 to this bit has no effect.
RERR1 bit sets to high when a read when empty error occurs. Software must clear this bit by writing 1. Writing 0 to this bit has no effect.
GAPOi. bit indicates that there has been a gap between transmission of two consecutive data from FIFO. This may happen when FIFO gets empty after state machine has sent a word and waits for the next. If control bit RPTLi is set to high this event will not occur. Software must clear this bit by writing 1. Writing 0 to this bit has no effect.
BERR sets to high when an error has occurred while writing to registers via APB. This may happen if the bus tries to write successively to same set of registers faster than the synchroniser block can cope with. Multiple switching may occur and contaminate the data during synchronisation. Software should clear this bit by writing 1. Writing 0 to this bit has no effect.
STAi bit indicates the current state of the channel which is useful for debugging purposes. 0 means the channel is not currently transmitting. 1 means channel is transmitting data.

Bit(s)	Field Name	Description	Type	Reset
31:13		***Reserved*** *- Write as 0, read as don't care*		
12	STA4	Channel 4 State	RW	0x0
11	STA3	Channel 3 State	RW	0x0
10	STA2	Channel 2 State	RW	0x0
9	STA1	Channel 1 State	RW	0x0
8	BERR	Bus Error Flag	RW	0x0
7	GAPO4	Channel 4 Gap Occurred Flag	RW	0x0
6	GAPO3	Channel 3 Gap Occurred Flag	RW	0x0
5	GAPO2	Channel 2 Gap Occurred Flag	RW	0x0
4	GAPO1	Channel 1 Gap Occurred Flag	RW	0x0
3	RERR1	Fifo Read Error Flag	RW	0x0
2	WERR1	Fifo Write Error Flag	RW	0x0

1	EMPT1	Fifo Empty Flag	RW	0x1
0	FULL1	Fifo Full Flag	RW	0x0

DMAC Register

Synopsis ENAB bit is used to start DMA.
PANIC bits are used to determine the threshold level for PANIC signal going active. Default value is 7.
DREQ bits are used to determine the threshold level for DREQ signal going active. Default value is 7.

Bit(s)	Field Name	Description	Type	Reset
31	ENAB	DMA Enable 0: DMA disabled 1: DMA enabled	RW	0x0
30:16		***Reserved*** *- Write as 0, read as don't care*		
15:8	PANIC	DMA Threshold for PANIC signal	RW	0x7
7:0	DREQ	DMA Threshold for DREQ signal	RW	0x7

RNG1 Register

Synopsis This register is used to define the range for the corresponding channel. In PWM mode evenly distributed pulses are sent within a period of length defined by this register. In serial mode serialised data is transmitted within the same period. If the value in PWM_RNGi is less than 32, only the first PWM_RNGi bits are sent resulting in a truncation. If it is larger than 32 excess zero bits are padded at the end of data. Default value for this register is 32.
Note: Channels 3 and 4 are not available in B0 and corresponding Channel Range Registers are ignored.

Bit(s)	Field Name	Description	Type	Reset
31:0	PWM_RNGi	Channel i Range	RW	0x20

DAT1 Register

Synopsis This register stores the 32 bit data to be sent by the PWM Controller when USEFi is 0. In PWM mode data is sent by pulse width modulation: the value of this register defines the number of pulses which is sent within the period defined by PWM_RNGi. In serialiser mode data stored in this register is serialised and transmitted.
Note: Channels 3 and 4 are not available in B0 and corresponding Channel Data Registers are ignored.

Bit(s)	Field Name	Description	Type	Reset
31:0	PWM_DATi	Channel i Data	RW	0x0

FIF1 Register

Synopsis This register is the FIFO input for the all channels. Data written to this address is stored in channel FIFO and if USEFi is enabled for the channel i it is used as data to be sent. This register is write only, and reading this register will always return bus default return value, pwm0 .
When more than one channel is enabled for FIFO usage, the data written into the FIFO is shared between these channels in turn. For example if the word series A B C D E F G H I .. is written to FIFO and two channels are active and configured to use FIFO then channel 1 will transmit words A C E G I .. and channel 2 will transmit words B D F H .. .
Note that requesting data from the FIFO is in locked-step manner and therefore requires tight coupling of state machines of the channels. If any of the channel range (period) value is different than the others this will cause the channels with small range values to wait between words hence resulting in gaps between words. To avoid that, each channel sharing the FIFO should be configured to use the same range value.
Also note that RPTLi are not meaningful when the FIFO is shared between channels as there is no defined channel to own the last data in the FIFO. Therefore sharing channels must have their RPTLi set to zero.
If the set of channels to share the FIFO has been modified after a configuration change, FIFO should be cleared before writing new data.

Bit(s)	Field Name	Description	Type	Reset
31:0	PWM_FIFO	Channel FIFO Input	RW	0x0

RNG2 Register

Synopsis This register is used to define the range for the corresponding channel. In PWM mode evenly distributed pulses are sent within a period of length defined by this register. In serial mode serialised data is transmitted within the same period. If the value in PWM_RNGi is less than 32, only the first PWM_RNGi bits are sent resulting in a truncation. If it is larger than 32 excess zero bits are padded at the end of data. Default value for this register is 32.
Note: Channels 3 and 4 are not available in B0 and corresponding Channel Range Registers are ignored.

Bit(s)	Field Name	Description	Type	Reset
31:0	PWM_RNGi	Channel i Range	RW	0x20

DAT2 Register

Synopsis This register stores the 32 bit data to be sent by the PWM Controller when USEFi is 1. In PWM mode data is sent by pulse width modulation: the value of this register defines the number of pulses which is sent within the period defined by PWM_RNGi. In serialiser mode data stored in this register is serialised and transmitted.
Note: Channels 3 and 4 are not available in B0 and corresponding Channel Data Registers are ignored.

Bit(s)	Field Name	Description	Type	Reset
31:0	PWM_DATi	Channel i Data	RW	0x0

10 SPI

10.1 Introduction

This Serial interface peripheral supports the following features:

- Implements a 3 wire serial protocol, variously called Serial Peripheral Interface (SPI) or Synchronous Serial Protocol (SSP).
- Implements a 2 wire version of SPI that uses a single wire as a bidirectional data wire instead of one for each direction as in standard SPI.
- Implements a LoSSI Master (Low Speed Serial Interface)
- Provides support for polled, interrupt or DMA operation.

10.2 SPI Master Mode

10.2.1 Standard mode

In Standard SPI master mode the peripheral implements the standard 3 wire serial protocol described below.

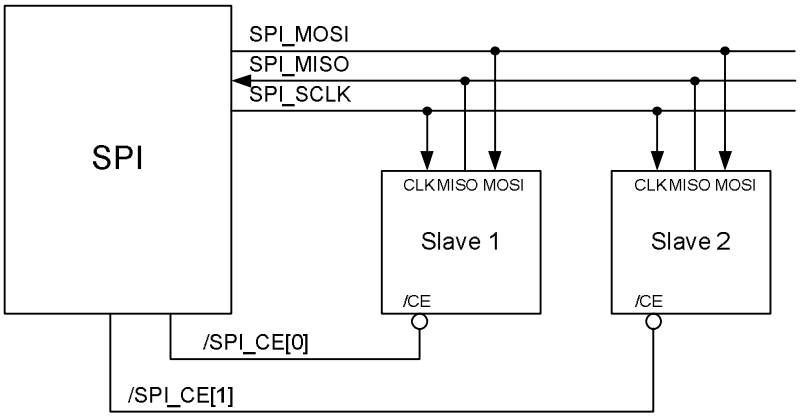
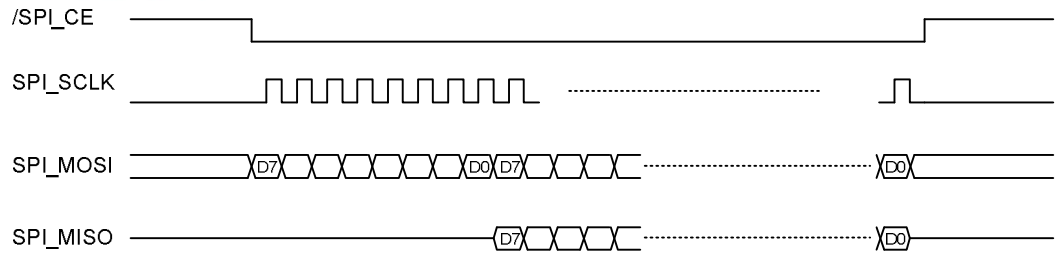


Figure 10-1 SPI Master Typical Usage



Notes
1. Slave enables itself onto SPI_MISO only when it is outputting data. At other times, output is tristate.
2. Different SCLK polarity and phase values are possible. Case illustrated is CPOL = 0, CPHA = 0.
3. Data is transmitted MSB first.
4. Transactions can be from a single byte to hundreds of bytes.

Figure 10-2 SPI Cycle

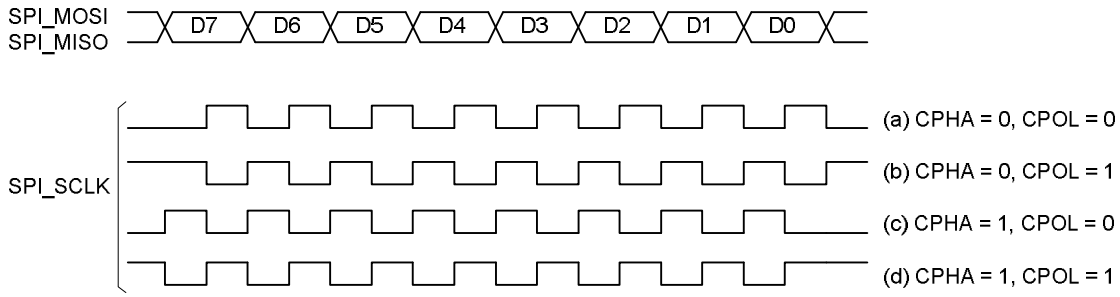


Figure 10-3 Different Clock Polarity/Phase

10.2.2 Bidirectional mode

In bidirectional SPI master mode the same SPI standard is implemented except that a single wire is used for the data (MIMO) instead of the two as in standard mode (MISO and MOSI). Bidirectional mode is used in a similar way to standard mode, the only difference is that before attempting to read data from the slave, you must set the read enable (SPI_REN) bit in the SPI control and status register (SPI_CS). This will turn the bus around, and when you write to the SPI_FIFO register (with junk) a read transaction will take place on the bus, and the read data will appear in the FIFO.

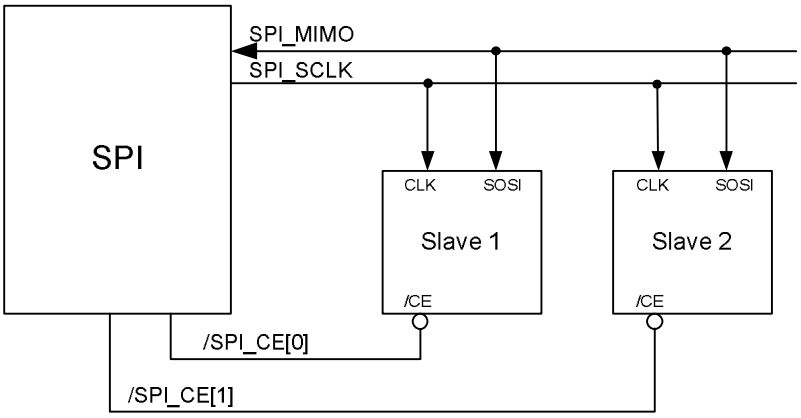


Figure 10-4 Bidirectional SPI Master Typical Usage

10.3 LoSSI mode

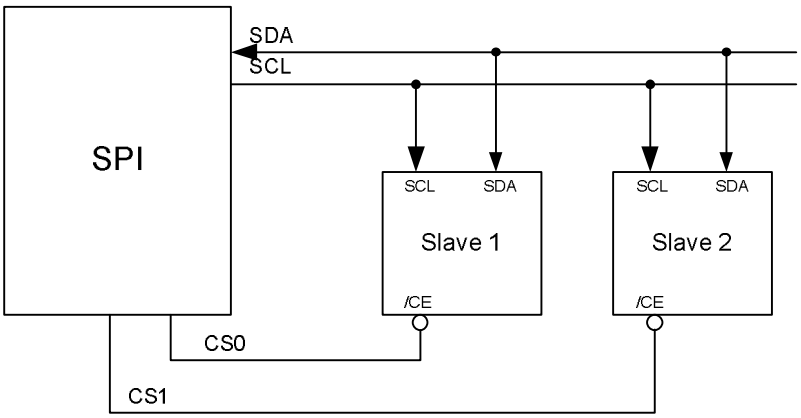
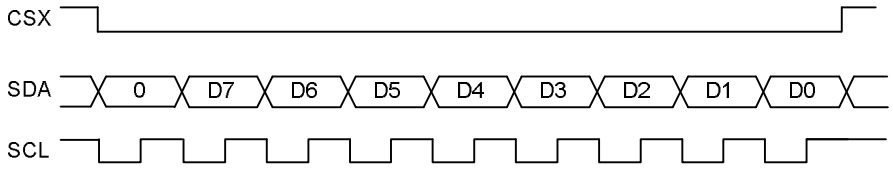


Figure 10-5 LoSSI mode Typical usage

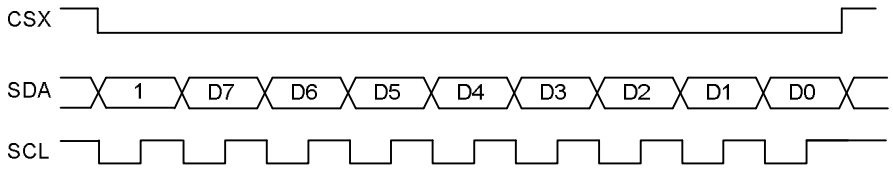
The LoSSI standard allows us to issue commands to peripherals and to transfer data to and from them. LoSSI commands and parameters are 8 bits long, but an extra bit is used to indicate whether the byte is a command or data. This extra bit is set high for a parameter and low for a command. The resulting 9-bit value is serialized to the output. When reading from a LoSSI peripheral the standard allows us to read bytes of data, as well as 24 and 32 bit words.

Commands and parameters are issued to a LoSSI peripheral by writing the 9-bit value of the command or data into the SPI_FIFO register as you would for SPI mode. Reads are automated in that if the serial interface peripheral detects a read command being issued, it will issue the command and complete the read transaction, putting the received data into the FIFO.

10.3.1 Command write

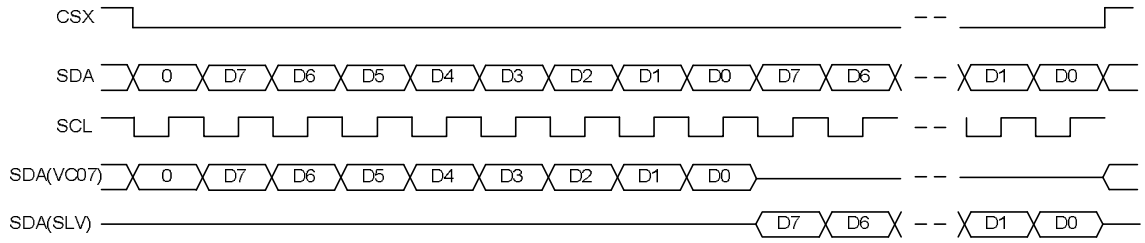


10.3.2 Parameter write



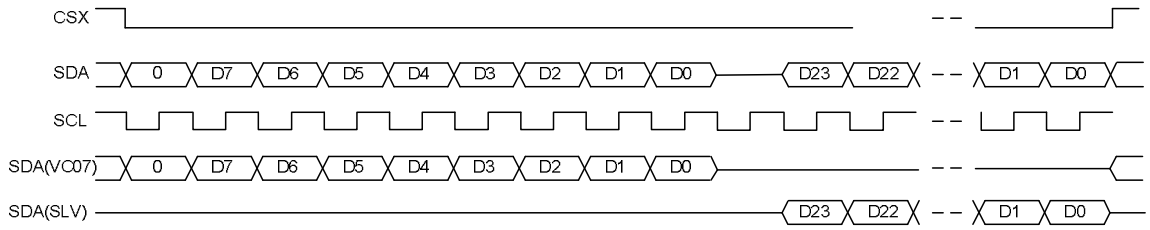
10.3.3 Byte read commands

Byte read commands are 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f, 0xda, 0xdb, 0xdc.



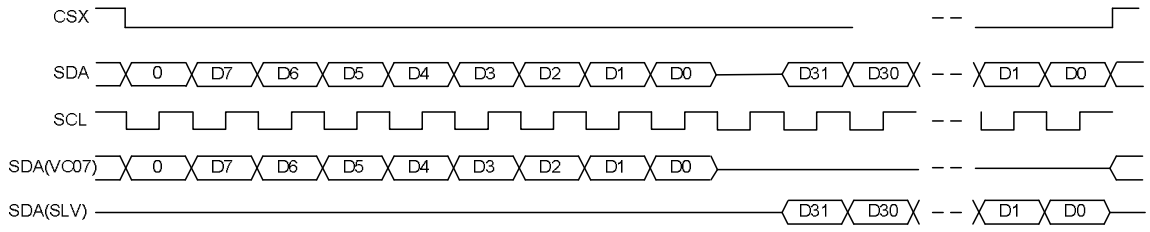
10.3.4 24bit read command

A 24 bit read can be achieved by using the command 0x04.



10.3.5 32bit read command

A 32bit read can be achieved by using the command 0x09.



10.4 Block Diagram

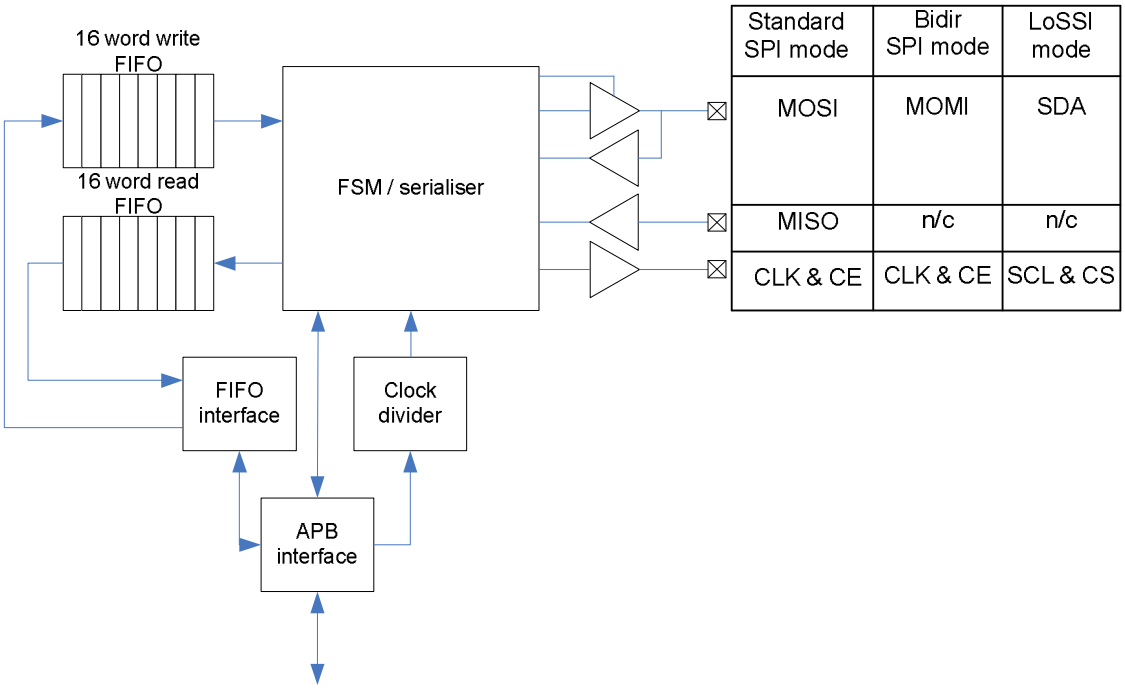


Figure 10-6 Serial interface Block Diagram

10.5 SPI Register Map

The BCM2835 devices has only one SPI interface of this type. It is referred to in all the documentation as SPI0. It has two additional mini SPI interfaces (SPI1 and SPI2). The specifiation of those can be found under *2.3 Universal SPI Master (2x)*.

The base address of this SPI0 interface is 0x7E204000.

SPI Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	CS	SPI Master Control and Status	32
0x4	FIFO	SPI Master TX and RX FIFOs	32
0x8	CLK	SPI Master Clock Divider	32

0xc	DLEN	SPI Master Data Length	32
0x10	LTOH	SPI LOSSI mode TOH	32
0x14	DC	SPI DMA DREQ Controls	32

CS Register

Synopsis This register contains the main control and status bits for the SPI.

Bit(s)	Field Name	Description	Type	Reset
31:26		***Reserved*** *- Write as 0, read as don't care*		
25	LEN_LONG	Enable Long data word in Lossi mode if DMA_LEN is set 0= writing to the FIFO will write a single byte 1= wrirng to the FIFO will write a 32 bit word	RW	0x0
24	DMA_LEN	Enable DMA mode in Lossi mode	RW	0x0
23	CSPOL2	Chip Select 2 Polarity 0= Chip select is active low. 1= Chip select is active high.	RW	0x0
22	CSPOL1	Chip Select 1 Polarity 0= Chip select is active low. 1= Chip select is active high.	RW	0x0
21	CSPOL0	Chip Select 0 Polarity 0= Chip select is active low. 1= Chip select is active high.	RW	0x0
20	RXF	RXF - RX FIFO Full 0 = RXFIFO is not full. 1 = RX FIFO is full. No further serial data will be sent/ received until data is read from FIFO.	RO	0x0
19	RXR	RXR RX FIFO needs Reading (full) 0 = RX FIFO is less than full (or not active TA = 0). 1 = RX FIFO is or more full. Cleared by reading sufficient data from the RX FIFO or setting TA to 0.	RO	0x0

18	TXD	TXD TX FIFO can accept Data 0 = TX FIFO is full and so cannot accept more data. 1 = TX FIFO has space for at least 1 byte.	RO	0x1
17	RXD	RXD RX FIFO contains Data 0 = RX FIFO is empty. 1 = RX FIFO contains at least 1 byte.	RO	0x0
16	DONE	Done transfer Done 0 = Transfer is in progress (or not active TA = 0). 1 = Transfer is complete. Cleared by writing more data to the TX FIFO or setting TA to 0.	RO	0x0
15	TE_EN	Unused	RW	0x0
14	LMONO	Unused	RW	0x0
13	LEN	LEN LoSSI enable The serial interface is configured as a LoSSI master. 0 = The serial interface will behave as an SPI master. 1 = The serial interface will behave as a LoSSI master.	RW	0x0
12	REN	REN Read Enable read enable if you are using bidirectional mode. If this bit is set, the SPI peripheral will be able to send data to this device. 0 = We intend to write to the SPI peripheral. 1 = We intend to read from the SPI peripheral.	RW	0x1
11	ADCS	ADCS Automatically Deassert Chip Select 0 = Don t automatically deassert chip select at the end of a DMA transfer chip select is manually controlled by software. 1 = Automatically deassert chip select at the end of a DMA transfer (as determined by SPIDLEN)	RW	0x0
10	INTR	INTR Interrupt on RXR 0 = Don t generate interrupts on RX FIFO condition. 1 = Generate interrupt while RXR = 1.	RW	0x0
9	INTD	INTD Interrupt on Done 0 = Don t generate interrupt on transfer complete. 1 = Generate interrupt when DONE = 1.	RW	0x0

8	DMAEN	DMAEN DMA Enable 0 = No DMA requests will be issued. 1 = Enable DMA operation. Peripheral generates data requests. These will be taken in four-byte words until the SPIDLEN has been reached.	RW	0x0
7	TA	Transfer Active 0 = Transfer not active./CS lines are all high (assuming CSPOL = 0). RXR and DONE are 0. Writes to SPIFIFO write data into bits -0 of SPICS allowing DMA data blocks to set mode before sending data. 1 = Transfer active. /CS lines are set according to CS bits and CSPOL. Writes to SPIFIFO write data to TX FIFO.TA is cleared by a dma_frame_end pulse from the DMA controller.	RW	0x0
6	CSPOL	Chip Select Polarity 0 = Chip select lines are active low 1 = Chip select lines are active high	RW	0x0
5:4	CLEAR	CLEAR FIFO Clear 00 = No action. x1 = Clear TX FIFO. One shot operation. 1x = Clear RX FIFO. One shot operation. If CLEAR and TA are both set in the same operation, the FIFOs are cleared before the new frame is started. Read back as 0.	RW	0x0
3	CPOL	Clock Polarity 0 = Rest state of clock = low. 1 = Rest state of clock = high.	RW	0x0
2	CPHA	Clock Phase 0 = First SCLK transition at middle of data bit. 1 = First SCLK transition at beginning of data bit.	RW	0x0
1:0	CS	Chip Select 00 = Chip select 0 01 = Chip select 1 10 = Chip select 2 11 = Reserved	RW	0x0

FIFO Register

Synopsis This register allows TX data to be written to the TX FIFO and RX data to be read from the RX FIFO.

Bit(s)	Field Name	Description	Type	Reset
31:0	DATA	DMA Mode (DMAEN set) If TA is clear, the first 32-bit write to this register will control SPIDLEN and SPICS. Subsequent reads and writes will be taken as four-byte data words to be read/written to the FIFOs Poll/Interrupt Mode (DMAEN clear, TA set) Writes to the register write bytes to TX FIFO. Reads from register read bytes from the RX FIFO	RW	0x0

CLK Register

Synopsis This register allows the SPI clock rate to be set.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		
15:0	CDIV	Clock Divider SCLK = Core Clock / CDIV If CDIV is set to 0, the divisor is 65536. The divisor must be a power of 2. Odd numbers rounded down. The maximum SPI clock rate is of the APB clock.	RW	0x0

DLEN Register

Synopsis This register allows the SPI data length rate to be set.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		
15:0	LEN	Data Length The number of bytes to transfer. This field is only valid for DMA mode (DMAEN set) and controls how many bytes to transmit (and therefore receive).	RW	0x0

LTOH Register

Synopsis This register allows the LoSSI output hold delay to be set.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3:0	TOH	This sets the Output Hold delay in APB clocks. A value of 0 causes a 1 clock delay.	RW	0x1

DC Register

Synopsis This register controls the generation of the DREQ and Panic signals to an external DMA engine The DREQ signals are generated when the FIFOs reach their defined levels and need servicing. The Panic signals instruct the external DMA engine to raise the priority of its AXI requests.

Bit(s)	Field Name	Description	Type	Reset
31:24	RPANIC	DMA Read Panic Threshold. Generate the Panic signal to the RX DMA engine whenever the RX FIFO level is greater than this amount.	RW	0x30
23:16	RDREQ	DMA Read Request Threshold. Generate A DREQ to the RX DMA engine whenever the RX FIFO level is greater than this amount, (RX DREQ is also generated if the transfer has finished but the RXFIFO isn t empty).	RW	0x20
15:8	TPANIC	DMA Write Panic Threshold. Generate the Panic signal to the TX DMA engine whenever the TX FIFO level is less than or equal to this amount.	RW	0x10
7:0	TDREQ	DMA Write Request Threshold. Generate a DREQ signal to the TX DMA engine whenever the TX FIFO level is less than or equal to this amount.	RW	0x20

10.6 Software Operation

10.6.1 Polled

a) Set CS, CPOL, CPHA as required and set TA = 1.

b) Poll TXD writing bytes to SPI_FIFO, RXD reading bytes from SPI_FIFO until all data written.

c) Poll DONE until it goes to 1.

d) Set TA = 0.

10.6.2 Interrupt

e) Set INTR and INTD. These can be left set over multiple operations.

f) Set CS, CPOL, CPHA as required and set TA = 1. This will immediately trigger a first interrupt with DONE == 1.

g) On interrupt:

h) If DONE is set and data to write (this means it is the first interrupt), write up to 16 bytes to SPI_FIFO. If DONE is set and no more data, set TA = 0. Read trailing data from SPI_FIFO until RXD is 0.

i) If RXR is set read 12 bytes data from SPI_FIFO and if more data to write, write up to 12 bytes to SPIFIFO.

10.6.3 DMA

Note: In order to function correctly, each DMA channel must be set to perform 32-bit transfers when communicating with the SPI. Either the Source or the Destination Transfer Width field in the DMA TI register must be set to 0 (i.e. 32-bit words) depending upon whether the channel is reading or writing to the SPI.

Two DMA channels are required, one to read from and one to write to the SPI.

j) Enable DMA DREQ's by setting the DMAEN bit and ADCS if required.

k) Program two DMA control blocks, one for each DMA controller.

l) DMA channel 1 control block should have its PER_MAP set to x and should be set to write 'transfer length' + 1 words to SPI_FIFO. The data should comprise:

   i) A word with the transfer length in bytes in the top sixteen bits, and the control register settings [7:0] in the bottom eight bits (i.e. TA = 1, CS, CPOL, CPHA as required.)

   ii) 'Transfer length' number in words of data to send.

m) DMA channel 2 control block should have its PER_MAP set to y and should be set to read 'transfer length' words from SPI_FIFO.

n) Point each DMA channel at its CB and set its ACTIVE bit to 1.

o) On receipt of an interrupt from DMA channel 2, the transfer is complete.

10.6.4 Notes

1. The SPI Master knows nothing of the peripherals it is connected to. It always both sends and receives bytes for every byte of the transaction.
2. SCLK is only generated during byte serial transfer. It pauses in the rest state if the next byte to send is not ready or RXF is set.
3. Setup and Hold times related to the automatic assertion and de-assertion of the CS lines when operating in DMA mode (DMAEN and ADCS set) are as follows:

   The CS line will be asserted at least 3 core clock cycles before the msb of the first byte of the transfer.

   The CS line will be de-asserted no earlier than 1 core clock cycle after the trailing edge of the final clock pulse.

   If these parameters are insufficient, software control should alleviate the problem. ADCS should be 0 allowing software to manually control the assertion and de-assertion of the CS lines.

11 SPI/BSC SLAVE

11.1 Introduction

The BSC interface can be used as either a Broadcom Serial Controller (BSC) or a Serial Peripheral Interface (SPI) controller. The BSC bus is a proprietary bus compliant with the Philips® I2C bus/interface version 2.1 January 2000. Both BSC and SPI controllers work in the slave mode. The BSC slave controller has specially built in the Host Control and Software Registers for a Chip booting. The BCS controller supports fast-mode (400Kb/s) and it is compliant to the I^2C bus specification version 2.1 January 2000 with the restrictions:

- I^2C slave only operation
- clock stretching is not supported
- 7-bit addressing only
-

There is only one BSC/SPI slave. The registers base addresses is 0x7E21_4000.

11.2 Registers

The SPI controller implements 3 wire serial protocol variously called Serial Peripheral Interface (SPI) or Synchronous Serial Protocol (SSP). BSC and SPI controllers do not have DMA connected, hence DMA is not supported.

I2C_SPI_SLV Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	DR	Data Register	32
0x4	RSR	The operation status register and error clear register	32
0x8	SLV	The I2C SPI Address Register holds the I2C slave address value	32
0xc	CR	The Control register is used to configure the I2C or SPI operation	32
0x10	FR	Flag register	32
0x14	IFLS	Interrupt fifo level select register	32
0x18	IMSC	Interupt Mask Set Clear Register	32

0x1c	RIS	Raw Interupt Status Register	32
0x20	MIS	Masked Interupt Status Register	32
0x24	ICR	Interupt Clear Register	32
0x28	DMACR	DMA Control Register	32
0x2c	TDR	FIFO Test Data	32
0x30	GPUSTAT	GPU Status Register	32
0x34	HCTRL	Host Control Register	32
0x38	DEBUG1	I2C Debug Register	32
0x3c	DEBUG2	SPI Debug Register	32

DR Register

Synopsis The I2C SPI Data Register is used to transfer/receive data characters and provide a Status and Flag information. Status and Flag information is also available via individual registers.

Bit(s)	Field Name	Description	Type	Reset
31:27	RXFLEVEL	RXFLEVEL RX FIFO Level Returns the current level of the RX FIFO use	RO	0x0
26:22	TXFLEVEL	TXFLEVEL TX FIFO Level Returns the current level of the TX FIFO use	RO	0x0
21	RXBUSY	RXBUSY Receive Busy 0 Receive operation inactive 1 Receive operation in operation	RO	0x0
20	TXFE	TXFE TX FIFO Empty 0 TX FIFO is not empty 1 When TX FIFO is empty	RO	0x1
19	RXFF	RXFE RX FIFO Full 0 FX FIFO is not full 1 When FX FIFO is full	RO	0x0

18	TXFF	TXFF TX FIFO Full 0 TX FIFO is not full 1 When TX FIFO is full	RO	0x0
17	RXFE	RXFE RX FIFO Empty 0 FX FIFO is not empty 1 When FX FIFO is empty	RO	0x1
16	TXBUSY	TXBUSY Transmit Busy 0 Transmit operation inactive 1 Transmit operation in operation	RO	0x0
15:10		***Reserved*** *- Write as 0, read as don't care*		
9	UE	TXUE TX Underrun Error 0 - No error case detected 1 Set when TX FIFO is empty and I2C master attempt to read a data character from I2C slave. Cleared by writing 0 to I2C SPI Status register .	RO	0x0
8	OE	RXOE RX Overrun Error 0 No error case detected 1 Set when RX FIFO is full and a new data character is received. Cleared by writing 0 to I2C SPI Status register .	RO	0x0
7:0	DATA	DATA Received/Transferred data characters Data written to this location is pushed into the TX FIFO. Data read from this location is fetched from the RX FIFO.	RW	0x0

RSR Register

Synopsis The operation status register and error clear register.

Bit(s)	Field Name	Description	Type	Reset
31:6		***Reserved*** *- Write as 0, read as don't care*		
5	RXDMABREQ	Unsupported, write zero, read as don't care	RO	0x0
4	RXDMAPREQ	Unsupported, write zero, read as don't care	RO	0x0
3	TXDMABREQ	Unsupported, write zero, read as don't care	RO	0x0

2	TXDMAPREQ	Unsupported, write zero, read as don't care	RO	0x0
1	UE	TXUE TX Underrun Error 0 - No error case detected 1 Set when TX FIFO is empty and I2C master attempt to read a data character from I2C slave. Cleared by writing 0 to it.	RW	0x0
0	OE	RXOE RX Overrun Error 0 No error case detected 1 Set when RX FIFO is full and a new data character is received. Cleared by writing 0 to it.	RW	0x0

SLV Register

Synopsis The I2C SPI Address Register holds the I2C slave address value. NOTE: It is of no use in SPI mode.

Bit(s)	Field Name	Description	Type	Reset
31:7		***Reserved*** *- Write as 0, read as don't care*		
6:0	ADDR	SLVADDR I2C Slave Address Programmable I2C slave address Note: In case HOSTCTRLEN bit is set from the I2C SPI Control Register bit SLVADDR[0] chooses the following: 0 - selects normal operation, i.e. accessing RX and TX FIFOs. 1 - selects access to I2C SPI SW Status Register or I2C SPI Host Control Register	RW	0x0

CR Register

Synopsis The Control register is used to configure the I2C or SPI operation.

Bit(s)	Field Name	Description	Type	Reset
31:14		***Reserved*** *- Write as 0, read as don't care*		

13	INV_TXF	INV-RX Inverse TX status flags 0 = default status flags When this bit is 0, bit 6 (TXFE - TX FIFO Empty) will reset to a 1 1 = inverted status flags When this bit is set, bit 6 (TXFE - TX FIFO Full) will reset to a 0 * Note: INV_TX bit changes the default values of 6 bit as it is specified for I2C SPI GPU Host Status Register .	RW	0x0
12	HOSTCTRLEN	HOSTCTRLEN Enable Control for Host 0 = Host Control disabled 1 = Host Control enabled Note: HOSTCTRLEN allows Host to request GPUSTAT or HCTRL register. The same behaviour is achieved from the GPU side using ENSTAT and ENCTRL.	RW	0x0
11	TESTFIFO	TESTFIFO TEST FIFO 0 = TESTT FIFO disabled 1 = TESTT FIFO enabled	RW	0x0
10	INV_RXF	INV-RX Inverse RX status flags 0 = default status flags When this bit is 0, bit 6 (RXFF - RX FIFO Full) will reset to a 0 1 = inverted status flags When this bit is 0, bit 6 (RXFF - RX FIFO Empty) will reset to a 1 * NOTE: INV_RX bit changes the default values of 7 bit as it is specified for I2C SPI GPU Host Status Register .	RW	0x0
9	RXE	RXE Receive Enable 0 = Receive mode disabled 1 = Receive mode enabled	RW	0x0
8	TXE	TXE Transmit Enable 0 = Transmit mode disabled 1 = Transmit mode enabled	RW	0x0
7	BRK	BRK Break current operation 0 = No effect. 1 = Stop operation and clear the FIFOs.	RW	0x0

6	ENCTRL	ENCTRL ENABLE CONTROL 8bit register 0 = Control register disabled. Implies ordinary I2C protocol. 1 = Control register enabled. When enabled the control register is received as a first data character on the I2C bus. NOTE: The same behaviour is achieved from the Host side by using bit SLVADDR[6] of the slave address.	RO	0x0
5	ENSTAT	ENSTAT ENABLE STATUS 8bit register 0 = Status register disabled. Implies ordinary I2C protocol. 1 = Status register enabled. When enabled the status register is transferred as a first data character on the I2C bus. Status register is transferred to the host. NOTE: The same behaviour is achieved from the Host side by using bit SLVADDR[6] of the slave address.	RW	0x0
4	CPOL	CPOL Clock Polarity 0 = 1 = SPI Related	RW	0x0
3	CPHA	CPHA Clock Phase 0 = 1 = SPI Related	RW	0x0
2	I2C	SPI Mode 0 = Disabled I2C mode 1 = Enabled I2C mode	RW	0x0
1	SPI	SPI Mode 0 = Disabled SPI mode 1 = Enabled SPI mode	RW	0x0
0	EN	EN Enable Device 1 = Enable I2C SPI Slave. 0 = Disable I2C SPI Slave.	RW	0x0

FR Register

Synopsis The flag register indicates the current status of the operation.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		

15:11	RXFLEVEL	RXFLEVEL RX FIFO Level Returns the current level of the RX FIFO use	RW	0x0
10:6	TXFLEVEL	TXFLEVEL TX FIFO Level Returns the current level of the TX FIFO use	RW	0x0
5	RXBUSY	RXBUSY Receive Busy 0 Receive operation inactive 1 Receive operation in operation	RW	0x0
4	TXFE	TXFE TX FIFO Empty 0 TX FIFO is not empty 1 When TX FIFO is empty	RW	0x1
3	RXFF	RXFE RX FIFO Full 0 FX FIFO is not full 1 When FX FIFO is full	RW	0x0
2	TXFF	TXFF TX FIFO Full 0 TX FIFO is not full 1 When TX FIFO is full	RW	0x0
1	RXFE	RXFE RX FIFO Empty 0 FX FIFO is not empty 1 When FX FIFO is empty	RW	0x1
0	TXBUSY	TXBUSY Transmit Busy 0 Transmit operation inactive 1 Transmit operation in operation	RW	0x0

IFLS Register

Synopsis The flag register indicates the current status of the operation.

Bit(s)	Field Name	Description	Type	Reset
31:12		***Reserved*** *- Write as 0, read as don't care*		
11:9	RXIFPSEL	Unsupported, write zero, read as don't care	RO	0x0
8:6	TXIFPSEL	Unsupported, write zero, read as don't care	RO	0x0

5:3	RXIFLSEL	RXIFLSEL RX Interrupt FIFO Level Select Interrupt is triggered when : 000 RX FIFO gets 1/8 full 001 RX FIFO gets 1/4 full 010 RX FIFO gets 1/2 full 011 RX FIFO gets 3/4 full 100 RX FIFO gets 7/8 full 101 111 not used	RW	0x0
2:0	TXIFLSEL	TXIFLSEL TX Interrupt FIFO Level Select Interrupt is triggered when : 000 TX FIFO gets 1/8 full 001 TX FIFO gets 1/4 full 010 TX FIFO gets 1/2 full 011 TX FIFO gets 3/4 full 100 TX FIFO gets 7/8 full 101 111 not used	RW	0x0

IMSC Register

Synopsis Interrupt Mask Set/Clear Register. On a read this register returns the current value of the mask on the relevant interrupt. On a write of 1 to the particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the corresponding mask.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3	OEIM	Overrun error interrupt mask. A read returns the current mask for the interrupt. On a write of 1, the mask of the OEINTR interrupt is set. A write of 0 clears the mask.	RW	0x0
2	BEIM	Break error interrupt mask. A read returns the current mask for the BEINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0
1	TXIM	Transmit interrupt mask. A read returns the current mask for the TXINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0
0	RXIM	Receive interrupt mask. A read returns the current mask for the RXINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0

RIS Register

Synopsis The Raw Interrupt Status Register returns the current raw status value, prior to masking, of the corresponding interrupt.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3	OERIS	Overrun error interrupt status. Returns the raw interrupt state of the OEINTR interrupt.	RW	0x0
2	BERIS	Break error interrupt status. Returns the raw interrupt state of the BEINTR interrupt.	RW	0x0
1	TXRIS	Transmit interrupt status. Returns the raw interrupt state of the TXINTR interrupt.	RW	0x0
0	RXRIS	Receive interrupt status. Returns the raw interrupt state of the RXINTR interrupt.	RW	0x0

MIS Register

Synopsis The Masked Interrupt Status Register returns the current masked status value of the corresponding interrupt.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3	OEMIS	Overrun error masked interrupt status. Returns the masked interrupt state of the OEINTR interrupt.	RW	0x0
2	BEMIS	Break error masked interrupt status. Returns the masked interrupt state of the BEINTR interrupt.	RW	0x0
1	TXMIS	Transmit masked interrupt status. Returns the masked interrupt state of the TXINTR interrupt.	RW	0x0
0	RXMIS	Receive masked interrupt status. Returns the masked interrupt state of the RXINTR interrupt.	RW	0x0

ICR Register

Synopsis The Interrupt Clear Register.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3	OEIC	Overrun error interrupt clear. Clears the OEINTR interrupt.	RW	0x0
2	BEIC	Break error interrupt clear. Clears the BEINTR interrupt.	RW	0x0
1	TXIC	Transmit interrupt clear. Clears the TXINTR interrupt.	RW	0x0
0	RXIC	Receive masked interrupt status. Returns the masked interrupt state of the RXINTR interrupt.	RW	0x0

DMACR Register

Synopsis The DMA Control register is not supported in this version.

Bit(s)	Field Name	Description	Type	Reset
31:3		***Reserved*** *- Write as 0, read as don't care*		
2	DMAONERR	Unsupported, write zero, read as don't care	RW	0x0
1	TXDMAE	Unsupported, write zero, read as don't care	RW	0x0
0	RXDMAE	Unsupported, write zero, read as don't care	RW	0x0

TDR Register

Synopsis The Test Data Register enables data to be written into the receive FIFO and read out from the transmit FIFO for test purposes.

Bit(s)	Field Name	Description	Type	Reset
31:8		***Reserved*** *- Write as 0, read as don't care*		
7:0	DATA	Test data is written into the receive FIFO and read out of the transmit FIFO.	RW	0x0

GPUSTAT Register

Synopsis The GPU SW Status Register to be passed via I2C bus to a Host.
NOTE: GPU SW Status Register is combined with the status bit coming from within I2C SPI Slave device. Hence, the I2C SPI GPU Host Status Register as it is seen by a Host is depicted on Table 1 14.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3:0	DATA	GPUSTAT GPU to Host Status Register SW controllable	RW	0x0

HCTRL Register

Synopsis The Host Control register is received from the host side via I2C bus. When ENCTRL - enable control register bit is set, the host control register is received as the first data character after the I2C address.

Bit(s)	Field Name	Description	Type	Reset
31:8		***Reserved*** *- Write as 0, read as don't care*		
7:0	DATA	HCTRL Host Control Register SW processing received via I2C bus	RW	0x0

DEBUG1 Register

Synopsis I2C Debug Register

Bit(s)	Field Name	Description	Type	Reset
31:26		***Reserved*** *- Write as 0, read as don't care*		
25:0	DATA		RW	0xe

DEBUG2 Register

Synopsis SPI Debug Register

Bit(s)	Field Name	Description	Type	Reset
31:24		***Reserved*** *- Write as 0, read as don't care*		
23:0	DATA		RW	0x400000

12 System Timer

The System Timer peripheral provides four 32-bit timer channels and a single 64-bit free running counter. Each channel has an output compare register, which is compared against the 32 least significant bits of the free running counter values. When the two values match, the system timer peripheral generates a signal to indicate a match for the appropriate channel. The match signal is then fed into the interrupt controller. The interrupt service routine then reads the output compare register and adds the appropriate offset for the next timer tick. The free running counter is driven by the timer clock and stopped whenever the processor is stopped in debug mode.

The Physical (hardware) base address for the system timers is 0x7E003000.

12.1 System Timer Registers

ST Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	CS	System Timer Control/Status	32
0x4	CLO	System Timer Counter Lower 32 bits	32
0x8	CHI	System Timer Counter Higher 32 bits	32
0xc	C0	System Timer Compare 0	32
0x10	C1	System Timer Compare 1	32
0x14	C2	System Timer Compare 2	32
0x18	C3	System Timer Compare 3	32

CS Register	
Synopsis	System Timer Control / Status. This register is used to record and clear timer channel comparator matches. The system timer match bits are routed to the interrupt controller where they can generate an interrupt. The M0-3 fields contain the free-running counter match status. Write a one to the relevant bit to clear the match detect status bit and the corresponding interrupt request line.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3	M3	System Timer Match 3 0 = No Timer 3 match since last cleared. 1 = Timer 3 match detected.	RW	0x0
2	M2	System Timer Match 2 0 = No Timer 2 match since last cleared. 1 = Timer 2 match detected.	RW	0x0
1	M1	System Timer Match 1 0 = No Timer 1 match since last cleared. 1 = Timer 1 match detected.	RW	0x0
0	M0	System Timer Match 0 0 = No Timer 0 match since last cleared. 1 = Timer 0 match detected.	RW	0x0

CLO Register

Synopsis System Timer Counter Lower bits.
The system timer free-running counter lower register is a read-only register that returns the current value of the lower 32-bits of the free running counter.

Bit(s)	Field Name	Description	Type	Reset
31:0	CNT	Lower 32-bits of the free running counter value.	RW	0x0

CHI Register

Synopsis System Timer Counter Higher bits.
The system timer free-running counter higher register is a read-only register that returns the current value of the higher 32-bits of the free running counter.

Bit(s)	Field Name	Description	Type	Reset
31:0	CNT	Higher 32-bits of the free running counter value.	RW	0x0

C0 C1 C2 C3 Register

Synopsis System Timer Compare.
The system timer compare registers hold the compare value for each of the four timer channels. Whenever the lower 32-bits of the free-running counter matches one of the compare values the corresponding bit in the system timer control/status register is set.
Each timer peripheral (minirun and run) has a set of four compare registers.

Bit(s)	Field Name	Description	Type	Reset
31:0	CMP	Compare value for match channel n.	RW	0x0

13 UART

The BCM2835 device has two UARTS. On mini UART and and PL011 UART. This section describes the PL011 UART. For details of the mini UART see *2.2 Mini UART*.

The PL011 UART is a Universal Asynchronous Receiver/Transmitter. This is the ARM UART (PL011) implementation. The UART performs serial-to-parallel conversion on data characters received from an external peripheral device or modem, and parallel-to-serial conversion on data characters received from the Advanced Peripheral Bus (APB).

The ARM PL011 UART has some optional functionality which can be included or left out.

The following functionality is ***not supported*** :

- Infrared Data Association (IrDA)
- Serial InfraRed (SIR) protocol Encoder/Decoder (ENDEC)
- Direct Memory Access (DMA).

The UART provides:

- Separate 16x8 transmit and 16x12 receive FIFO memory.
- Programmable baud rate generator.
- Standard asynchronous communication bits (start, stop and parity). These are added prior to transmission and removed on reception.
- False start bit detection.
- Line break generation and detection.
- Support of the modem control functions CTS and RTS. However DCD, DSR, DTR, and RI are not supported.
- Programmable hardware flow control.
- Fully-programmable serial interface characteristics:

  data can be 5, 6, 7, or 8 bits

  even, odd, stick, or no-parity bit generation and detection

  1 or 2 stop bit generation

  baud rate generation, dc up to UARTCLK/16

The UART clock source and associated dividers are controlled by the Clock Manager.

For the in-depth UART overview, please, refer to the ARM PrimeCell UART (PL011) Revision: r1p5 Technical Reference Manual.

13.1 Variations from the 16C650 UART

The UART varies from the industry-standard 16C650 UART device as follows:

- Receive FIFO trigger levels are 1/8, 1/4, 1/2, 3/4, and 7/8
- Transmit FIFO trigger levels are 1/8, 1/4, 1/2, 3/4, and 7/8
- The internal register map address space, and the bit function of each register differ

- The deltas of the modem status signals are not available.

The following 16C650 UART features are not supported:

- 1.5 stop bits (1 or 2 stop bits only are supported)
- Independent receive clock.

13.2 Primary UART Inputs and Outputs

The UART has two primary inputs RXD, nCTS and two primary outputs TXD, nRTS. The remaining signals like SRIN, SROUT, OUT1, OUT2, DSR, DTR, and RI are not supported in this implementation. The following table shows the UART signals map on the General Purpose I/O (GPIO). For the insight on how to program alternate function refer to the GPIO paragraph.

	Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
GPIO14	Low	TXD0					
GPIO15	Low	RXD0					
GPIO16	Low				CTS0		
GPIO17	Low				RTS0		
GPIO30	Low				CTS0		
GPIO31	Low				RTS0		
GPIO32	Low				TXD0		
GPIO33	Low				RXD0		
GPIO36	High			TXD0			
GPIO37	Low			RXD0			
GPIO38	Low			RTS0			
GPIO39	Low			CTS0			

Table 13-1 UART Assignment on the GPIO Pin map

13.3 UART Interrupts

The UART has one intra-chip interrupt UARTINTR generated as the OR-ed function of the five individual interrupts.

• UARTINTR, this is an OR function of the five individual masked outputs:

- UARTRXINTR
- UARTTXINTR
- UARTRTINTR
- UARTMSINTR, that can be caused by:
  - — UARTCTSINTR, because of a change in the nUARTCTS modem status
  - — UARTDSRINTR, because of a change in the nUARTDSR modem status.
- UARTEINTR, that can be caused by an error in the reception:

— UARTOEINTR, because of an overrun error

— UARTBEINTR, because of a break in the reception

— UARTPEINTR, because of a parity error in the received character

— UARTFEINTR, because of a framing error in the received character.

One can enable or disable the individual interrupts by changing the mask bits in the Interrupt Mask Set/Clear Register, UART_IMSC. Setting the appropriate mask bit HIGH enables the interrupt.

UARTRXINTR:

The transmit interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO is equal to or lower than the programmed trigger level then the transmit interrupt is asserted HIGH. The transmit interrupt is cleared by writing data to the transmit FIFO until it becomes greater than the trigger level, or by clearing the interrupt.
- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the transmit interrupt is asserted HIGH. It is cleared by performing a single write to the transmit FIFO, or by clearing the interrupt.

UARTRTINTR:

The receive interrupt changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO reaches the programmed trigger level. When this happens, the receive interrupt is asserted HIGH. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than the trigger level, or by clearing the interrupt.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the receive interrupt is asserted HIGH. The receive interrupt is cleared by performing a single read of the receive FIFO, or by clearing the interrupt.

13.4 Register View

The PL011 USRT is mapped on base adderss 0x7E20100. It has the following memory-mapped registers.

UART Address Map			
Address Offset	**Register Name**	**Description**	**Size**
0x0	DR	Data Register	32
0x4	RSRECR		32
0x18	FR	Flag register	32

0x20	ILPR	not in use	32
0x24	IBRD	Integer Baud rate divisor	32
0x28	FBRD	Fractional Baud rate divisor	32
0x2c	LCRH	Line Control register	32
0x30	CR	Control register	32
0x34	IFLS	Interupt FIFO Level Select Register	32
0x38	IMSC	Interupt Mask Set Clear Register	32
0x3c	RIS	Raw Interupt Status Register	32
0x40	MIS	Masked Interupt Status Register	32
0x44	ICR	Interupt Clear Register	32
0x48	DMACR	DMA Control Register	32
0x80	ITCR	Test Control register	32
0x84	ITIP	Integration test input reg	32
0x88	ITOP	Integration test output reg	32
0x8c	TDR	Test Data reg	32

DR Register

Synopsis The UART_DR Register is the data register. For words to be transmitted:
if the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO.
if the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO).
The write operation initiates transmission from the UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted.
For received words:
if the FIFOs are enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO
if the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO).

Bit(s)	Field Name	Description	Type	Reset
31:12		***Reserved*** *- Write as 0, read as don't care*		
11	OE	Overrun error. This bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it.	RW	0x0
10	BE	Break error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received.	RW	0x0
9	PE	Parity error. When set to 1, it indicates that the parity of the received data character does not match the parity that the EPS and SPS bits in the Line Control Register, UART_LCRH select. In FIFO mode, this error is associated with the character at the top of the FIFO.	RW	0x0

8	FE	Framing error. When set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO.	RW	0x0
7:0	DATA	Receive (read) data character. Transmit (write) data character.	RW	0x0

RSRECR Register

Synopsis The UART_RSRECR Register is the receive status register/error clear register. If the status is read from this register, then the status information for break, framing and parity corresponds to the data character read from the Data Register, UART_DR. The status information for overrun is set immediately when an overrun condition occurs. NOTE: The received data character must be read first from the Data Register, UART_DR on before reading the error status associated with that data character from this register.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3	OE	Overrun error. This bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it.	RW	0x0

2	BE	Break error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received.	RW	0x0
1	PE	Parity error. When set to 1, it indicates that the parity of the received data character does not match the parity that the EPS and SPS bits in the Line Control Register, UART_LCRH select. In FIFO mode, this error is associated with the character at the top of the FIFO.	RW	0x0
0	FE	Framing error. When set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO.	RW	0x0

FR Register

Synopsis The UART_FR Register is the flag register.

Bit(s)	Field Name	Description	Type	Reset
31:9		***Reserved*** *- Write as 0, read as don't care*		
8	RI	Unsupported, write zero, read as don't care	RW	0x0

7	TXFE	Transmit FIFO empty. The meaning of this bit depends on the state of the FEN bit in the Line Control Register, UARTLCR_ LCRH. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty. This bit does not indicate if there is data in the transmit shift register.	RW	0x1
6	RXFF	Receive FIFO full. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_ LCRH Register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, the RXFF bit is set when the receive FIFO is full.	RW	0x0
5	TXFF	Transmit FIFO full. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_ LCRH Register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, the TXFF bit is set when the transmit FIFO is full.	RW	0x0
4	RXFE	Receive FIFO empty. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H Register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, the RXFE bit is set when the receive FIFO is empty.	RW	0x0
3	BUSY	UART busy. If this bit is set to 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty, regardless of whether the UART is enabled or not.	RW	0x0
2	DCD	Unsupported, write zero, read as don't care	RW	0x0
1	DSR	Unsupported, write zero, read as don't care	RW	0x0

0	CTS	Clear to send. This bit is the complement of the UART clear to send, nUARTCTS, modem status input. That is, the bit is 1 when nUARTCTS is LOW.	RW	0x0

ILPR Register

Synopsis This is the disabled IrDA register, writing to it has not effect and reading returns 0.

Bit(s)	Field Name	Description	Type	Reset
31:0	ILPR	Reserved - write zero, read as don't care.	RW	0x0

IBRD Register

Synopsis The UART_IBRD Register is the integer part of the baud rate divisor value.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		
15:0	IBRD	The integer baud rate divisor.	RW	0x0

FBRD Register

Synopsis The UART_FBRD Register is the fractional part of the baud rate divisor value. The baud rate divisor is calculated as follows:
Baud rate divisor BAUDDIV = (FUARTCLK/(16 Baud rate))
where FUARTCLK is the UART reference clock frequency. The BAUDDIV is comprised of the integer value IBRD and the fractional value FBRD. NOTE: The contents of the IBRD and FBRD registers are not updated until transmission or reception of the current character is complete.

Bit(s)	Field Name	Description	Type	Reset
31:6		***Reserved*** *- Write as 0, read as don't care*		
5:0	FBRD	The fractional baud rate divisor.	RW	0x0

LCRH Register

Synopsis The UARTLCR_ LCRH Register is the line control register.
NOTE: The UART_LCRH, UART_IBRD, and UART_FBRD registers must not be changed:
when the UART is enabled
when completing a transmission or a reception when it has been programmed to become disabled.

Bit(s)	Field Name	Description	Type	Reset
31:8		***Reserved*** *- Write as 0, read as don't care*		
7	SPS	Stick parity select. 0 = stick parity is disabled 1 = either: if the EPS bit is 0 then the parity bit is transmitted and checked as a 1 if the EPS bit is 1 then the parity bit is transmitted and checked as a 0. See Table 25 9.	RO	0x0
6:5	WLEN	Word length. These bits indicate the number of data bits transmitted or received in a frame as follows: b11 = 8 bits b10 = 7 bits b01 = 6 bits b00 = 5 bits.	RW	0x0
4	FEN	Enable FIFOs: 0 = FIFOs are disabled (character mode) that is, the FIFOs become 1-byte-deep holding registers 1 = transmit and receive FIFO buffers are enabled (FIFO mode).	RW	0x0

3	STP2	Two stop bits select. If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.	RW	0x0
2	EPS	Even parity select. Controls the type of parity the UART uses during transmission and reception: 0 = odd parity. The UART generates or checks for an odd number of 1s in the data and parity bits. 1 = even parity. The UART generates or checks for an even number of 1s in the data and parity bits. This bit has no effect when the PEN bit disables parity checking and generation. See Table 25 9.	RW	0x0
1	PEN	Parity enable: 0 = parity is disabled and no parity bit added to the data frame 1 = parity checking and generation is enabled. See Table 25 9.	RW	0x0
0	BRK	Send break. If this bit is set to 1, a low-level is continually output on the TXD output, after completing transmission of the current character.	RW	0x0

CR Register

Synopsis The UART_CR Register is the control register.
NOTE:
To enable transmission, the TXE bit and UARTEN bit must be set to 1. Similarly, to enable reception, the RXE bit and UARTEN bit, must be set to 1.
NOTE:
Program the control registers as follows:
1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by setting the FEN bit to 0 in the Line Control Register, UART_LCRH.
4. Reprogram the Control Register, UART_CR.
5. Enable the UART.

Bit(s)	Field Name	Description	Type	Reset
31:16		***Reserved*** *- Write as 0, read as don't care*		
15	CTSEN	CTS hardware flow control enable. If this bit is set to 1, CTS hardware flow control is enabled. Data is only transmitted when the nUARTCTS signal is asserted.	RW	0x0
14	RTSEN	RTS hardware flow control enable. If this bit is set to 1, RTS hardware flow control is enabled. Data is only requested when there is space in the receive FIFO for it to be received.	RW	0x0
13	OUT2	Unsupported, write zero, read as don't care	RO	0x0
12	OUT1	Unsupported, write zero, read as don't care	RO	0x0
11	RTS	Request to send. This bit is the complement of the UART request to send, nUARTRTS, modem status output. That is, when the bit is programmed to a 1 then nUARTRTS is LOW.	RW	0x0
10	DTR	Unsupported, write zero, read as don't care	RO	0x0
9	RXE	Receive enable. If this bit is set to 1, the receive section of the UART is enabled. Data reception occurs for UART signals. When the UART is disabled in the middle of reception, it completes the current character before stopping.	RW	0x1
8	TXE	Transmit enable. If this bit is set to 1, the transmit section of the UART is enabled. Data transmission occurs for UART signals. When the UART is disabled in the middle of transmission, it completes the current character before stopping.	RW	0x1
7	LBE	Loopback enable. If this bit is set to 1, the UARTTXD path is fed through to the UARTRXD path. In UART mode, when this bit is set, the modem outputs are also fed through to the modem inputs. This bit is cleared to 0 on reset, to disable loopback.	RW	0x0

6:3		***Reserved*** *- Write as 0, read as don't care*		
2	SIRLP	Unsupported, write zero, read as don't care	RO	0x0
1	SIREN	Unsupported, write zero, read as don't care	RO	0x0
0	UARTEN	UART enable: 0 = UART is disabled. If the UART is disabled in the middle of transmission or reception, it completes the current character before stopping. 1 = the UART is enabled.	RW	0x0

IFLS Register

Synopsis The UART_IFLS Register is the interrupt FIFO level select register. You can use this register to define the FIFO level that triggers the assertion of the combined interrupt signal.
The interrupts are generated based on a transition through a level rather than being based on the level. That is, the interrupts are generated when the fill level progresses through the trigger level.
The bits are reset so that the trigger level is when the FIFOs are at the half-way mark.

Bit(s)	Field Name	Description	Type	Reset
31:12		***Reserved*** *- Write as 0, read as don't care*		
11:9	RXIFPSEL	Unsupported, write zero, read as don't care	RO	0x0
8:6	TXIFPSEL	Unsupported, write zero, read as don't care	RO	0x0
5:3	RXIFLSEL	Receive interrupt FIFO level select. The trigger points for the receive interrupt are as follows: b000 = Receive FIFO becomes 1/8 full b001 = Receive FIFO becomes 1/4 full b010 = Receive FIFO becomes 1/2 full b011 = Receive FIFO becomes 3/4 full b100 = Receive FIFO becomes 7/8 full b101-b111 = reserved.	RW	0x0

2:0	TXIFLSEL	Transmit interrupt FIFO level select. The trigger points for the transmit interrupt are as follows: b000 = Transmit FIFO becomes 1/8 full b001 = Transmit FIFO becomes 1/4 full b010 = Transmit FIFO becomes 1/2 full b011 = Transmit FIFO becomes 3/4 full b100 = Transmit FIFO becomes 7/8 full b101-b111 = reserved.	RW	0x0

IMSC Register

Synopsis The UART_IMSC Register is the interrupt mask set/clear register. It is a read/write register. On a read this register returns the current value of the mask on the relevant interrupt. On a write of 1 to the particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the corresponding mask.

Bit(s)	Field Name	Description	Type	Reset
31:11		***Reserved*** *- Write as 0, read as don't care*		
10	OEIM	Overrun error interrupt mask. A read returns the current mask for the interrupt. On a write of 1, the mask of the UARTOEINTR interrupt is set. A write of 0 clears the mask.	RW	0x0
9	BEIM	Break error interrupt mask. A read returns the current mask for the UARTBEINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0
8	PEIM	Parity error interrupt mask. A read returns the current mask for the UARTPEINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0
7	FEIM	Framing error interrupt mask. A read returns the current mask for the UARTFEINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0

6	RTIM	Receive timeout interrupt mask. A read returns the current mask for the UARTRTINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0
5	TXIM	Transmit interrupt mask. A read returns the current mask for the UARTTXINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0
4	RXIM	Receive interrupt mask. A read returns the current mask for the UARTRXINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0
3	DSRMIM	Unsupported, write zero, read as don't care	RO	0x0
2	DCDMIM	Unsupported, write zero, read as don't care	RO	0x0
1	CTSMIM	nUARTCTS modem interrupt mask. A read returns the current mask for the UARTCTSINTR interrupt. On a write of 1, the mask of the interrupt is set. A write of 0 clears the mask.	RW	0x0
0	RIMIM	Unsupported, write zero, read as don't care	RO	0x0

RIS Register

Synopsis The UART_RIS Register is the raw interrupt status register. It is a read-only register. This register returns the current raw status value, prior to masking, of the corresponding interrupt.
NOTE: All the bits, except for the modem status interrupt bits (bits 3 to 0), are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

Bit(s)	Field Name	Description	Type	Reset
31:11		***Reserved*** *- Write as 0, read as don't care*		

10	OERIS	Overrun error interrupt status. Returns the raw interrupt state of the UARTOEINTR interrupt.	RW	0x0
9	BERIS	Break error interrupt status. Returns the raw interrupt state of the UARTBEINTR interrupt.	RW	0x0
8	PERIS	Parity error interrupt status. Returns the raw interrupt state of the UARTPEINTR interrupt.	RW	0x0
7	FERIS	Framing error interrupt status. Returns the raw interrupt state of the UARTFEINTR interrupt.	RW	0x0
6	RTRIS	Receive timeout interrupt status. Returns the raw interrupt state of the UARTRTINTR interrupt.	RW	0x0
5	TXRIS	Transmit interrupt status. Returns the raw interrupt state of the UARTTXINTR interrupt.	RW	0x0
4	RXRIS	Receive interrupt status. Returns the raw interrupt state of the UARTRXINTR interrupt.	RW	0x0
3	DSRRMIS	Unsupported, write zero, read as don't care	RW	0x0
2	DCDRMIS	Unsupported, write zero, read as don't care	RW	0x0
1	CTSRMIS	nUARTCTS modem interrupt status. Returns the raw interrupt state of the UARTCTSINTR interrupt.	RW	0x0
0	RIRMIS	Unsupported, write zero, read as don't care	RW	0x0

MIS Register

Synopsis The UART_MIS Register is the masked interrupt status register. This register returns the current masked status value of the corresponding interrupt. NOTE: All the bits, except for the modem status interrupt bits (bits 3 to 0), are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

Bit(s)	Field Name	Description	Type	Reset
31:11		***Reserved*** *- Write as 0, read as don't care*		
10	OEMIS	Overrun error masked interrupt status. Returns the masked interrupt state of the UARTOEINTR interrupt.	RW	0x0
9	BEMIS	Break error masked interrupt status. Returns the masked interrupt state of the UARTBEINTR interrupt.	RW	0x0
8	PEMIS	Parity error masked interrupt status. Returns the masked interrupt state of the UARTPEINTR interrupt.	RW	0x0
7	FEMIS	Framing error masked interrupt status. Returns the masked interrupt state of the UARTFEINTR interrupt.	RW	0x0
6	RTMIS	Receive timeout masked interrupt status. Returns the masked interrupt state of the UARTRTINTR interrupt.	RW	0x0
5	TXMIS	Transmit masked interrupt status. Returns the masked interrupt state of the UARTTXINTR interrupt.	RW	0x0
4	RXMIS	Receive masked interrupt status. Returns the masked interrupt state of the UARTRXINTR interrupt.	RW	0x0
3	DSRMMIS	Unsupported, write zero, read as don't care	RW	0x0
2	DCDMMIS	Unsupported, write zero, read as don't care	RW	0x0
1	CTSMMIS	nUARTCTS modem masked interrupt status. Returns the masked interrupt state of the UARTCTSINTR interrupt.	RW	0x0
0	RIMMIS	Unsupported, write zero, read as don't care	RW	0x0

ICR Register

Synopsis The UART_ICR Register is the interrupt clear register.

Bit(s)	Field Name	Description	Type	Reset
31:11		***Reserved*** *- Write as 0, read as don't care*		
10	OEIC	Overrun error interrupt clear. Clears the UARTOEINTR interrupt.	RW	0x0
9	BEIC	Break error interrupt clear. Clears the UARTBEINTR interrupt.	RW	0x0
8	PEIC	Parity error interrupt clear. Clears the UARTPEINTR interrupt.	RW	0x0
7	FEIC	Framing error interrupt clear. Clears the UARTFEINTR interrupt..	RW	0x0
6	RTIC	Receive timeout interrupt clear. Clears the UARTRTINTR interrupt.	RW	0x0
5	TXIC	Transmit interrupt clear. Clears the UARTTXINTR interrupt.	RW	0x0
4	RXIC	Receive masked interrupt status. Returns the masked interrupt state of the UARTRXINTR interrupt.	RW	0x0
3	DSRMIC	Unsupported, write zero, read as don't care	RW	0x0
2	DCDMIC	Unsupported, write zero, read as don't care	RW	0x0
1	CTSMIC	nUARTCTS modem masked interrupt status. Returns the masked interrupt state of the UARTCTSINTR interrupt.	RW	0x0
0	RIMIC	Unsupported, write zero, read as don't care	RW	0x0

DMACR Register

Synopsis This is the disabled DMA Control Register, writing to it has not effect and reading returns 0.

Bit(s)	Field Name	Description	Type	Reset
31:3		***Reserved*** *- Write as 0, read as don't care*		
2	DMAONERR	Unsupported, write zero, read as don't care	RW	0x0
1	TXDMAE	Unsupported, write zero, read as don't care	RW	0x0
0	RXDMAE	Unsupported, write zero, read as don't care	RW	0x0

ITCR Register

Synopsis This is the Test Control Register UART_ITCR.

Bit(s)	Field Name	Description	Type	Reset
31:2		***Reserved*** *- Write as 0, read as don't care*		
1	ITCR1	Test FIFO enable. When this bit it 1, a write to the Test Data Register, UART_DR writes data into the receive FIFO, and reads from the UART_DR register reads data out of the transmit FIFO. When this bit is 0, data cannot be read directly from the transmit FIFO or written directly to the receive FIFO (normal operation).	RW	0x0
0	ITCR0	Integration test enable. When this bit is 1, the UART is placed in integration test mode, otherwise it is in normal operation.	RW	0x0

ITIP Register

Synopsis This is the Test Control Register UART_ITIP.

Bit(s)	Field Name	Description	Type	Reset
31:4		***Reserved*** *- Write as 0, read as don't care*		
3	ITIP3	Reads return the value of the nUARTCTS primary input.	RW	0x0
2:1		***Reserved*** *- Write as 0, read as don't care*		
0	ITIP0	Reads return the value of the UARTRXD primary input.	RW	0x0

ITOP Register

Synopsis This is the Test Control Register UART_ITOP.

Bit(s)	Field Name	Description	Type	Reset
31:12		***Reserved*** *- Write as 0, read as don't care*		
11	ITOP11	Intra-chip output. Writes specify the value to be driven on UARTMSINTR. Reads return the value of UARTMSINTR at the output of the test multiplexor.	RW	0x0
10	ITOP10	Intra-chip output. Writes specify the value to be driven on UARTRXINTR. Reads return the value of UARTRXINTR at the output of the test multiplexor.	RW	0x0
9	ITOP9	Intra-chip output. Writes specify the value to be driven on UARTTXINTR. Reads return the value of UARTTXINTR at the output of the test multiplexor.	RW	0x0

8	ITOP8	Intra-chip output. Writes specify the value to be driven on UARTRTINTR. Reads return the value of UARTRTINTR at the output of the test multiplexor.	RW	0x0
7	ITOP7	Intra-chip output. Writes specify the value to be driven on UARTEINTR. Reads return the value of UARTEINTR at the output of the test multiplexor.	RW	0x0
6	ITIP6	Intra-chip output. Writes specify the value to be driven on UARTINTR. Reads return the value of UARTINTR at the output of the test multiplexor.	RW	0x0
5:4		***Reserved*** *- Write as 0, read as don't care*		
3	ITIP3	Primary output. Writes specify the value to be driven on nUARTRTS.	RW	0x0
2:1		***Reserved*** *- Write as 0, read as don't care*		
0	ITIP0	Primary output. Writes specify the value to be driven on UARTTXD.	RW	0x0

TDR Register

Synopsis UART_TDR is the test data register. It enables data to be written into the receive FIFO and read out from the transmit FIFO for test purposes. This test function is enabled by the ITCR1bit in the Test Control Register, UART_ITCR.

Bit(s)	Field Name	Description	Type	Reset
31:11		***Reserved*** *- Write as 0, read as don't care*		
10:0	TDR10_0	When the ITCR1 bit is set to 1, data is written into the receive FIFO and read out of the transmit FIFO.	RW	0x0

14 Timer (ARM side)

14.1 Introduction

The ARM Timer is *based* on a ARM AP804, but it has a number of differences with the standard SP804:

- There is only one timer.
- It only runs in continuous mode.
- It has a extra clock pre-divider register.
- It has a extra stop-in-debug-mode control bit.
- It also has a 32-bit free running counter.

The clock from the ARM timer is derived from the system clock. This clock can change dynamically e.g. if the system goes into reduced power or in low power mode. Thus the clock speed adapts to the overal system performance capabilities. For accurate timing it is recommended to use the system timers.

14.2 Timer Registers:

The base address for the ARM timer register is 0x7E00B000.

Address offset[8]	Description
0x400	Load
0x404	Value (Read Only)
0x408	Control
0x40C	IRQ Clear/Ack (Write only)
0x410	RAW IRQ (Read Only)
0x414	Masked IRQ (Read Only)
0x418	Reload
0x41C	*Pre-divider (Not in real 804!)*
0x420	*Free running counter (Not in real 804!)*

Timer Load register

The timer load register sets the time for the timer to count down. This value is loaded into the timer value register after the load register has been written or if the timer-value register has counted down to 0.

[8] This is the offset which needs to be added to the base address to get the full hardware address.

Timer Value register:

This register holds the current timer value and is counted down when the counter is running. It is counted down each timer clock until the value 0 is reached. Then the value register is re-loaded from the timer load register and the interrupt pending bit is set. The timer count down speed is set by the timer pre-divide register.

Timer control register:

The standard SP804 timer control register consist of 8 bits but in the BCM implementation there are more control bits for the extra features. Control bits 0-7 are identical to the SP804 bits, albeit some functionality of the SP804 is not implemented. All new control bits start from bit 8 upwards. Differences between a real 804 and the BCM implementation are shown in italics.

Name: Timer control		**Address: base + 0x40C**	**Reset: 0x3E0020**
Bit(s)	**R/W**	**Function**	
31:10	-	<Unused>	
23:16	*R/W*	*Free running counter pre-scaler. Freq is sys_clk/(prescale+1)* ***These bits do not exists in a standard 804! Reset value is 0x3E***	
15:10	-	<Unused>	
9	*R/W*	*0 : Free running counter Disabled* *1 : Free running counter Enabled* ***This bit does not exists in a standard 804 timer!***	
8	*R/W*	*0 : Timers keeps running if ARM is in debug halted mode* *1 : Timers halted if ARM is in debug halted mode* ***This bit does not exists in a standard 804 timer!***	
7	R/W	0 : Timer disabled 1 : Timer enabled	
6	*R/W*	***Not used**, The timer is always in free running mode.* *If this bit is set it enables periodic mode in a standard 804. That mode is not supported in the BC2835M.*	
5	R/W	0 : Timer interrupt disabled 1 : Timer interrupt enabled	
4	R/W	<Not used>	
3:2	R/W	Pre-scale bits: 00 : pre-scale is clock / 1 (No pre-scale) 01 : pre-scale is clock / 16 10 : pre-scale is clock / 256 11 : pre-scale is clock / 1 *(Undefined in 804)*	
1	R/W	0 : 16-bit counters 1 : 23-bit counter	
0	*R/W*	***Not used**, The timer is always in wrapping mode.* *If this bit is set it enables one-shot mode in real 804. That mode is not supported in the BCM2835.*	

Timer IRQ clear register:

The timer IRQ clear register is write only. When writing this register the interrupt-pending bit is cleared.

When reading this register it returns 0x544D5241 which is the ASCII reversed value for "ARMT".

Timer Raw IRQ register

The raw IRQ register is a read-only register. It shows the status of the interrupt pending bit.

Name: Raw IRQ		Address: base + 0x40C	Reset: 0x3E0020
Bit(s)	**R/W**	**Function**	
31:0	R	0	
0	R	0 : The interrupt pending bits is clear 1 : The interrupt pending bit is set.	

The interrupt pending bits is set each time the value register is counted down to zero. The interrupt pending bit can not by itself generates interrupts. Interrupts can only be generated if the interrupt enable bit is set.

Timer Masked IRQ register:

The masked IRQ register is a read-only register. It shows the status of the interrupt signal. It is simply a logical AND of the interrupt pending bit and the interrupt enable bit.

Name: Masked IRQ		Address: base + 0x40C	Reset: 0x3E0020
Bit(s)	**R/W**	**Function**	
31:0	R	0	
0	R	0 : Interrupt line not asserted. 1 :Interrupt line is asserted, (the interrupt pending and the interrupt enable bit are set.)	

Timer Reload register:

This register is a copy of the timer load register. The difference is that a write to this register does not trigger an immediate reload of the timer value register. Instead the timer load register value is only accessed if the value register has finished counting down to zero.

The timer pre-divider register:

Name: pre-divide		Address: base + 0x41C	Reset: 0x07D
Bit(s)	**R/W**	**Function**	
31:10	-	<Unused>	
9:0	R/W	Pre-divider value.	

The Pre-divider register is not present in the SP804.

The pre-divider register is 10 bits wide and can be written or read from. This register has been added as the SP804 expects a 1MHz clock which we do not have. Instead the pre-divider takes the APB clock and divides it down according to:

timer_clock = apb_clock/(pre_divider+1)

The reset value of this register is 0x7D so gives a divide by 126.

Free running counter

Name: Free running		Address: base + 0x420	Reset: 0x000
Bit(s)	**R/W**	**Function**	
31:0	R	Counter value	

The free running counter is not present in the SP804.

The free running counter is a 32 bits wide read only register. The register is enabled by setting bit 9 of the Timer control register. The free running counter is incremented immediately after it is enabled. The timer can not be reset but when enabled, will always increment and roll-over. The free running counter is also running from the APB clock and has its own clock pre-divider controlled by bits 16-23 of the timer control register.

This register will be halted too if bit 8 of the control register is set and the ARM is in Debug Halt mode.

15 USB

The USB core used in the Videocore is build from Synopsys IP. Details about the block can be found in DWC_otg_databook.pdf (Which can also be downloaded from https://www.synopsys.com/dw/ipdir.php?ds=dwc_usb_2_0_hs_otg) .

15.1 Configuration

A number of features of the block are specified before the block is build and thus can not be changed using software. The above mentioned document has a list of these under the chapter "Configuration Parameters". The following table list all configuration parameters mentioned in that chapter and the values which have been chosen.

Feature/Parameter	Selected value
Mode of Operation	0: HNP- and SRP-Capable OTG (Device and Host)
LPM Mode of Operation	0: Non-LPM-capable core
HSIC Mode of Operation	0: Non-HSIC-capable core
Architecture	2: Internal DMA
Point-to-Point Application Only	0: No
High-Speed PHY Interfaces	1: UTMI+
USB 1.1 Full-Speed Serial Transceiver Interface	1: Dedicated FS
USB IC_USB Transceiver Interface	0: Non-IC_USB-capable
Default (Power on) Interface selection: FS_USB/IC_USB	0 : FS_USB interface
Data Width of the UTMI+ Interface	0: 8 bits
Enable I2C Interface	0: None
Enable ULPI Carkit	0: No
Enable PHY Vendor Control Interface	0: No

Feature/Parameter	Selected value
Number of Device Mode Endpoints in Addition to Control Endpoint 0	7
Enable Dedicated Transmit FIFOs for Device IN Endpoints	1: Yes
Enable descriptor based scatter/gather DMA	0: No
Enable Option for Endpoint- Specific Interrupt	0: No
Number of Device Mode Periodic IN Endpoints	0
Number of Device Mode IN Endpoints including Control Endpo 0	8
Number of Device Mode Control Endpoints in Addition to Endpoint 0	0
Number of Host Mode Channels	8
Is Periodic OUT Channel Support Needed in Host Mode	1: Yes
Total Data FIFO RAM Depth	4096
Enable Dynamic FIFO Sizing	1: Yes
Largest Rx Data FIFO Depth	4096
Largest Non-Periodic Host Tx Data FIFO Depth	1024
Largest Non-periodic Tx Data FIFO Depth	4096
Largest Host Mode Tx Periodic Data FIFO Dept	4096
Non-periodic Request Queue Depth	8
Host Mode Periodic Request Queue Depth	8
Device Mode IN Token Sequence Learning Queue Depth	8
Width of Transfer Size Counters	19

Feature/Parameter	Selected value
Width of Packet Counters	10
Remove Optional Features	0: No
Power-on Value of User ID Register	0x2708A000
Enable Power Optimization	0: No
Is Minimum AHB Operating Frequency Less than 60 MHz	1: Yes
Reset Style of Clocked always Blocks in RTL	0: Asynchronous
Instantiate Double- Synchronization Flops	1: Yes
Enable Filter on “iddig” Signal from PHY	1: Yes
Enable Filter on “vbus_valid” Signal from PHY	1: Yes
Enable Filter on “a_valid” Signal from PHY	1: Yes
Enable Filter on “b_valid” Signal from PHY	1: Yes
Enable Filter on “session_end” Signal from PHY	1: Yes
Direction of Endpoints	Mode is {IN and OUT} for all endpoints
Largest Device Mode Periodic Tx Data FIFO n Depth	768 for all endpoints (Except 0)
Largest Device Mode IN Endpoint Tx FIFOn Depth (n = 0 to 15) when using dynamic FIFO sizing	0=32 1..5=512 6,7=768

15.2 Extra / Adapted registers.

Besides the registers as specified in the documentation of Synopsys a number of extra registers have been added. These control the Analogue USB Phy and the connections of the USB block into the Video core bus structure. Also the USB_GAHBCFG register has an alternative function for the bits [4:1].

Base Address of the USB block – 0x7E98_0000

Offset Address	Description		Size	Read/ Write
0x080	USB_MDIO_CNTL	MDIO interface control		R/W
0x084	USB_MDIO_GEN	Data for MDIO interface	32	R/W
0x088	USB_VBUS_DRV	Vbus and other Miscellaneous controls		R/W

USB MDIO Control (USB_MDIO_CNTL)

Address 0x 7E98 0080

Bit Number	Field Name	Description	Read/ Write	Reset
31	mdio_busy	1= MDIO read or write in progress 0= MDIO Idle	R	0
30-24	-	Unused	-	0
23	bb_mdo	Direct write (bitbash) MDO output	R/W	0
22	bb_mdc	Direct write (bitbash) MDC output	R/W	0
21	bb_enbl	1= MDIO bitbash enable 0= MDIO under control of the phy	R/W	0
20	freerun	1= MDC is continous active 0 = MDC only active during data transfer	R/W	0
19:16	mdc_ratio	MDC clock freq is sysclk/mdc_ratio	R/W	0
15:0	mdi	16-bit read of MDIO input shift register. Updates on falling edge of MDC	RO	0

Table 15-1 MDIO Control

USB MDIO Data (USB_MDIO_DATA)

Address 0x 7E98 0084

Bit Number	Field Name	Description	Read/ Write	Reset
31-0	mdio_data	32-bit sequence to send over MDIO bus	W	0
31-0	mdio_data	32-bit sequence received from MDIO bus	R	0

Table 15-2 USB MDIO data

A Preamble is not auto-generated so any MDIO access must be preceded by a write to this register of 0xFFFFFFFF. Furthermore, a bug in the USB PHY requires an extra clock edge so a write of 0x00000000 must follow the actual access.

USB VBUS (USB_VBUS)

Address 0x 7E98 0088

Bit Number	Field Name	Description	Read/ Write	Reset
31-20	-	Unused	-	0
19-16	axi_priority	Sets the USB AXI priority level	R/W	0
15:10	-	Unused	-	0
9	vbus_irq	1=one or more bits of [6:4] have changed since last read. This bit is cleared when the register is read.	RC	0
8	vbus_irq_en	1=Enable IRQ on VBUS status change	R/W	0
7	afe_non_driving	1=USB PHY AFE pull ups/pull downs are off 0=Normal USB AFE operation (Has no effect if MDIO mode is enabled in the phy)	R/W	0
6	utmisrp_dischrgvbus	Drive VBUS	R	0
5	utmisrp_chrgvbus	Charge VBUS	R	0
4	utmiotg_drvvbus	Discharge VBUS	R	0
3	utmiotg_avalid	A session Valid	R/W	0
2	utmiotg_bvalid	B session Valid	R/W	0
1	utmiotg_vbusvalid	VBUS valid	R/W	0
0	utmisrp_sessend	Session end	R/W	0

Table 15-3 USB MDIO data

The RW bits in this register are fed into the USB2.0 controller and the RO bits are coming out of it. In the real device, it will be up to the software to communicate this information between the USB2.0 controller and external VBUS device (some of these have I2C control, others will have to interface via GPIO).

USB AHB configuration (USB_GAHBCFG)

Address 0x 7E98 0008

The USB_GAHBCFG register has been adapted. Bits [4:1] which are marked in the Synopsys documentation as "Burst Length/Type (HBstLen)" have been used differently.

[4] 1 = Wait for all outstanding AXI writes to complete before signalling (internally) that DMA is done.
0 = don't wait.

[3] Not used

[2:1] Sets the maximum AXI burst length, but the bits are inverted,
00 = maximum AXI burst length of 4,
01 = maximum AXI burst length of 3,
10 = maximum AXI burst length of 2
11 = maximum AXI burst length of 1

Personal Notes:

APPENDIX C: CARBONTRACK SMART GATEWAY SPECIFICATIONS

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

Smart Gateway

Energy Management & IoT Gateway

Intelligent energy system

With carbonTRACK, you are in control. Welcome to the world of smart energy, where you can monitor, control and optimize how you use, store and share energy – all via an app or online dashboard.

Know your energy usage

Know how much energy you use and produce in real-time. Avoid bill-shock and use carbonTRACK app's bill prediction and comparisons, set bill goals and alerts to implement changes. Control and automate energy usage and save big.

Make the most of your solar system

Rooftop solar systems are a big investment. carbonTRACK shows your solar production, use, and export, and can notify you a fault does occur. For greater ROI for your solar, schedule appliances and circuits to run when the sun is shining and use excess energy to charge your battery or export to grid.

Be future-ready

With a carbonTRACK Smart Gateway, your energy ecosystem can be adjusted based on energy prices and your home automation needs. Control multiple IoT devices from the carbonTRACK app, schedule and automate devices inline to your energy usage, change your preferences in line with your evolving lifestyle.

BENEFITS

- Monitor how much energy your home or business is consuming.
- Control electrical circuits and individual appliances through scheduling or by switching them on/off through the smartphone app or online dashboard.
- Monitor your solar power generation, usage and export.
- Always know your system status with intuitive real-time alerts.

FEATURES

- App & online dashboard
- Connectivity to SunSpec enabled devices
- Switch & smart control
- Reliable, real-time data
- Solar & battery maximisation
- Control multiple IoT devices
- ZigBee HA 1.2 compliant
- Google Home connectivity
- Safety & system alerts
- MDT data security

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

SPECIFICATIONS

Model	CT200-CAT-M1	CT200-3G
Performance	LTE CAT-M1 / NB-IoT Power Class 3 +23dBm	HSDPA, UMTS, EDGE, GPRS (Class 12), GSM
Frequency Bands	Full Spectrum Radio	GPRS/EDGE: 1900/1800/900/850MHz WCDMA: 2100/850MHz
Processor & Memory	• 8-bit AVR Micro-controller with 131 instructions • 12 MHz (External Crystal) • 128KB Flash Memory • 16KB Data Memory • 4KB EEPROM	
Radio Frequency ZigBee Power and Standard	2.4GHz Channels 11-26, +10dBm, ZigBee HA 1.2	
Radio Frequency Wi-Fi and Power	802.11 b/g/n with WPA/WPA2, 2.4-2.4835 GHz Channels 1-14, +20 dBm	
Monitoring Accuracy	2% +/- @ PF 1.0 to 0.6	
Input Voltage and Current	100 – 240 V; 200 mA Maximum (Optional; 5 to 24 V DC)	
Battery Type, Capacity and Run Time	Lithium ion, 3.7 V/ 1500 mAh, 5.55 Wh, 24 hours	
Firmware Upgrade Method	Remote Over the Air (OTA) Firmware Upgradable	
Connectors		
Monitoring Current Clamps	x4 (30A to 3000A each)	
Circuit Relay Controls	x3 (30A each)	
Cellular Antenna	Internal Antennal Cell 2dBi, Optional external antenna via Female SMA	
Wi-Fi Antenna	Internal Antenna, Optional external antenna via Female SMA	
SIM Socket	SIM / USIM (2FF)	
Ethernet Port	10/100 MB Ethernet	
RS-485 Port	Supports Half-duplex and Full-duplex	
Physical Description		
Dimensions (W x L x H)	7.98” x 10.53” x 2.36” (202.73 mm x 267.45 mm x 60.00 mm)	
Weight & Chassis Type	1.54 lbs (0.70 kg), Plastic	
Environmental		
Operating Temperature	-20° to +55° C*	
Storage Temperature	-40° to +85° C	
Relative Humidity	20% to 90%, non-condensing	
Warranty		
Warranty Term	12 months	
Certifications		
EMC Compliance	SANS 222 / CISPR 22, SANS / IEC 61000-3-3, SANS / IEC 61000-4-2, SANS / IEC 61000-4-3, SANS / IEC 61000-4-4, SANS / IEC 61000-4-5, SANS / IEC 61000-4-6, SANS / IEC 61000-4-8, SANS / IEC 61000-4-11	
Radio Compliance	ACMA Section 376 of the Telecommunications Act 1997, AS/CA S042.1: 2011, AS/ACIF S042.3: 2005, AS/NZS 60950.1: 2011, ESTI EN 301 908-2 V5.4.1 (2012-12), ESTI EN 301 908-2 V6.2.1 (2013-04)	
Safety Compliance	ANSI/UL 60950-1-2014, CSA/CAN C22.2 No. 60950-1-07, IEC60950-1 / SANS 60950-1, IEC61010-1	
Environmental Compliance	SANS 60529: 2013 Ed 1.2/IEC 60529: 2013 Ed2.2	

* Installation in outdoor locations or ambient temperature above 40° C or 70° C has not been evaluated by UL. UL Certification does not apply or extend to use in outdoor applications. Certification does not apply or extend to voltages outside certified range and has not been evaluated by UL for operating voltages beyond tested range.

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

GENERAL INFORMATION

Functional Capabilities	
Digital Essential Loads	• Create and manage digital essential loads panel
Green Circuit	• Supports green circuit feature to maximize solar use
Peak Load Management	• Manage peak loads for an entire home
Device Control	• Remote control and monitoring of all appliances in the home
User Configurable Schedules	• User configurable schedule-based control
Data Transmission Protocol	• Secure MDT protocol
Electrical Parameters	
Electrical Phase Monitoring	• Single and three phases (3 phase 4 wire)
Power Direction	• Measures both import and export
Solar Support	• Yes (shareable among total of 3 current clamps)
Monitoring	
Active Power	• Both Watt and Watt-hour in all phases
Apparent Power	• Both VA and VAh in all phases
Power Factor	• Yes (0 to 1 PF) in all phases
Power Direction	• Measures import, export and net powers (kWh)
Product Compatibility	
Electric Hot Water Unit	• Support up to 2 electrical hot water/geysers • Manual control / configurable switch on timers • Peak load timer for efficient heating
Solar Thermal Controller	• Temperature differential based control • Circulation Pump switching based on user configurable upper and lower limits
HVAC (Heating, Venting, and Air-Conditioning)	• Support up to 3 HVAC systems, Temperature based control
Solenoid Valve	• Shut-off valve control for hot water units
ZigBee capability	• On-board ZigBee support
Future Expansion (wired)	• Any device supporting I2C or RS232 communication
Future Expansion (wireless)	• ZigBee router implementation
Relay Control	• Enabled through three (3) onboard relays
Relay Port Voltage and Rated Current	• 12 V DC, 1 A
Relay Output Voltage and Rated Current	• 240 VAC, 30 A
Wireless Device Control	• Enabled through ZigBee
External Sensors	
Voltage Sensor and Type	• 100 V to 240 V line-to-neutral, on-board
Current Sensor and Type	• 60 A standard sensor (60 A to 2000 A options), off-board
Temperature Sensor	• 2 Analog and up to 8 digital, -55 °C to 125 °C
Water Flow Sensor	• 1 Litre/min to 60 Litre/min
Leakage Sensor	• Water leak detector, alert triggered
Earth Fault Alarm	• Supports electrical earth fault leakage

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

ORDERING INFORMATION

SKU	Description	Region
CT200M-508	CT200 Smart Gateway (CAT-M1 Modem)	All
CT200i-505	CT200 Smart Gateway (3G Modem)	All

RECOMMENDED ACCESSORIES

SKU	Description	Region
CT20E2-010	Outdoor Weatherproof Enclosure	All
AN10D1-010	10dBi External Antenna	All
AN01M1-014	1m Antenna Extension Cable	All
AN05M1-016	5m Antenna Extension Cable	All
AN10M1-202	10m Antenna Extension Cable	All
CCJ011-100	Insulated Cable Clamp 60 Amp	All
CCJ121-101	Insulated Cable Clamp 120 Amp	All
CCJ201-102	Insulated Cable Clamp 200 Amp	All
TSA2M1-205	Temperature Sensor - Analogue - 20M - MTC10K	All
TSD201-215	Temperature Probe - Digital - 20M	All
CB20A1-042	Single Pole MCB 20 Amp/Bypass Switch - Internal	All
CB25A1-043	Single Pole MCB 25 Amp	All
CTSMPL-750	Smart Plug	All

Go to www.carbonTRACK.com.au for detailed product model numbers.

SERVICE & WARRANTY

carbonTRACK's comprehensive Support Service programs offer a full array of options to suit your specific needs. These services are aimed at protecting your investment and reducing the total cost of ownership.

TECHNICAL SUPPORT SERVICES

At carbonTRACK, we are committed to providing you with a quality service with a high attention to detail. Several options of support are available to choose from.

For additional information on Support Services as well as other service offerings, please contact your carbonTRACK representative or visit www.carbonTRACK.com.au

HEAD OFFICE
carbonTRACK Limited
Level 2
104 Burwood Road
Hawthorn, Victoria
Australia 3122
Tel: 1300 288 648
Email: info@carbontrack.com.au
www.carbonTRACK.com.au

APPENDIX D: CARBONTRACK SMARTPLUG SPECIFICATIONS

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

carbonTRACK Smart Plug

IoT Monitoring & Control

Pinpoint energy hungry appliances

Unsure of what's using the most electricity in your home? Plug in one appliance at a time and track how much each appliance consumes. Then you can see if it's time for an upgrade, or if you need it checked out for a service.

Run appliances, even when you're away

Use your cheaper electricity to wash or dry your clothes. Connect your appliances and switch them on even when you're away from home. Clean and dry clothes with less effort, and less costs!

Secure your home from intruders

Going away over the weekend or heading overseas for a holiday? Connect your lamps to a smart plug. Schedule lights to come on at night and create the illusion of activity.

Manage your phantom loads

Connect your appliances to smart plugs and switch them off from your phone when you're about to leave the house or go to sleep.

BENEFITS

- See how much electricity your appliances use, in 15-minute intervals
- Switch connected appliances remotely from your phone
- Save your time and reduce wasted electricity
- Shift your appliance schedules and run them when electricity is cheaper

FEATURES

- App & online dashboard
- Big Picture & granular data
- Switch & smart control
- Reliable, real-time data
- ZigBee HA 1.2 compliant
- MDT data security

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

SPECIFICATIONS

Model	Smart Plug (AUS/NZ)	Smart Plug (USA)
Radio Frequency ZigBee Power and Standard	2.4GHz Channels 11-26, +10dBm, ZigBee HA 1.2	
Input Voltage	100 – 240 V	
Maximum Load Current @ 110 volts	Resistive: 10A Inductive: 10A	Resistive: 15A Inductive: 10A
Maximum Load Current @ 220 volts	Resistive: 10A Inductive: 10A	Resistive: 10A Inductive: 6A
Operating Power	Running Load: < 0.7 Watts; Standby: < 0.7 Watts	
Metering Accuracy	Better than 2% 2W~1500W	
Maximum Working Range (Line of Sight) *	15 metres	
Firmware Upgrade Method	Not Applicable	
Physical Description		
Dimensions (W x L x H)	4.01” x 2.52” x 1.50” (102.00 mm x 64.00 mm x 38.00 mm)	
Weight	0.29 lbs (0.13 kg)	
Chassis Type	Plastic	
Environmental		
Operating Temperature	-20° to +55° C	
Storage Temperature	-40° to +85° C	
Relative Humidity	20% to 90%, non-condensing	
Certifications		
Conformance	Conforms to UL STD 60730-1:2016 and 498A:2008	
Certification	Certified to CSA STD E60730-1:2015 and C22.2#42:2010	

GENERAL INFORMATION

Model	Smart Plug (AUS/NZ)	Smart Plug (USA)
Functions	• Remote on/off control, ideal for home appliance control	
Schedules	• User configurable schedule-based control	
Data Transmission Protocol	• Secure carbonTRACK MDT protocol	
Monitoring		
Power	• Measures Watts and Watt-Hours	
Control Capabilities		
Switching	• Manual and scheduled switching ON/OFF	

*Note: There are various factors that can affect the maximum working range of this device such as obstructions between the Smart Hub and the Smart Plug. The maximum working range is an estimate only and the manufacturer make no guarantees that this range will be achieved for your specific application.

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

ORDERING INFORMATION

SKU	Description	Region
CTSMPL-750	Smart Plug (AUS/NZ)	All
CTSMPL-751	Smart Plug (USA)	All

RECOMMENDED ACCESSORIES

SKU	Description	Region
CT200M-508	CT200 Smart Gateway (CAT-M1 Modem)	All
CT200i-505	CT200 Smart Gateway (3G Modem)	All

Go to www.carbonTRACK.com.au for detailed product model numbers.

SERVICE & WARRANTY

carbonTRACK's comprehensive Support Service programs offer a full array of options to suit your specific needs. These services are aimed at protecting your investment and reducing the total cost of ownership.

TECHNICAL SUPPORT SERVICES

At carbonTRACK, we are committed to providing you with a quality service with a high attention to detail. Several options of support are available to choose from.

For additional information on Support Services as well as other service offerings, please contact your carbonTRACK representative or visit www.carbonTRACK.com.au

HEAD OFFICE
carbonTRACK Limited
Level 2
104 Burwood Road
Hawthorn, Victoria
Australia 3122
Tel: 1300 288 648
Email: info@carbontrack.com.au
www.carbonTRACK.com.au

APPENDIX E: CARBONTRACK SMART THERMOSTAT ESPECIFICATIONS

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

Smart Thermostat

HVAC Monitoring and Control

Control your Heating and Cooling

The Smart Thermostat device is an all in one monitoring and control module for HVAC systems, encapsulated in a small form factor and can be controlled and configured remotely via the carbonTRACK Smart Hub.

Flexibility

The Smart Thermostat device is a very smart, adaptable control device.

It can be used with a number of leading HVAC systems currently available on the market.

Compatibility

The Smart Thermostat supports a range of 2 Heat / 2 Cool multistage conventional HVAC and Heat Pump systems.

BENEFITS

- Control and manage your HVAC temperature and running schedule
- Installation is a simple replacement of your existing HVAC controller. No change to the main HVAC unit is required.

FEATURES

- Control your device from the App & online dashboard
- Switch & smart control
- Device scheduling
- Reliable, real-time data
- ZigBee HA 1.2 compliant
- Safety & system alerts
- MDT data security

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

SPECIFICATIONS

Model	PCTZ-503
Radio Frequency ZigBee Power and Standard	2.4GHz +8dBm, ZigBee HA 1.2
Device Range (Outdoor / Indoor)	100 metres / 30 metres*
Electrical Rating	24 VAC, 2A Carry; 5A Surge 50/60 Hz
Rated Power Consumption	4-watts maximum
Firmware Upgrade Method	Not Supported
Mounting Type	Wall mounted
Physical Description	
Dimensions (W x L x H)	87.40 mm x 160.00 mm x 33.00 mm
Weight	0.226 kg
Chassis Type	Plastic
Environmental	
Operating Temperature	-0° to +40° C
Storage Temperature	-30° to +60° C
Relative Humidity	20% to 90%, non-condensing
Visual Characteristics	
Display	3.5” TFT Colour LCD, 480*320 pixels
Certifications	
FCC Certification	Yes
CE Certification	Yes

GENERAL INFORMATION

Model	PCTZ-503
Functions	• 2-stage heater and 2-stage cooling HVAC systems • Supports natural gas, heat pump, electric, hot water, steam or gravity, gas fireplaces (24 Volts), oil heat sources • Supports any combination of systems
Heating and Cooling	• Up to 2 stages cooling and Up to 2 stages heating
Heat Pump	• Up to 2 Stages • 1 stage + 1 aux • 1 stage + 2 aux • 2 stage + 1 aux • 2 stage + 2 aux • Dual Fuel Heat Pump • 4 stage heating (2 compressor, 2 aux heat), 2 stage cooling
Schedules	• User configurable schedule-based control
Data Transmission Protocol	• Secure carbonTRACK MDT protocol
HVAC Control Functions	
System Mode	• Heat, Cool, Auto, Off, Emergency Heat (Heat Pump only)
Fan Mode	• On, Auto, Circulation
Advanced Mode	• Local and remote setting of the temperature • Auto-changeover between heat and cool mode (System Auto) • Compressor short cycle protection delay of 2 minutes • Failure protection by cutting off all circuit relays
Temperature Sensing Range	• -10° to +125° C
Temperature Resolution	• 0.1° C
Temperature Display Accuracy	• +/- 1.0° C
Temperature Setpoint Span	• +/- 1.0° C
Humidity Sensing Range	• 0 to 100% relative humidity
Humidity Accuracy	• +/- 4.0% Accuracy through the range of 0% to 80% relative humidity
Humidity Response Time	• 18 seconds to reach 63% of the next step value

*Note: There are various factors that can affect the maximum working range of this device such as obstructions between the Smart Hub and the Smart Thermostat. The maximum working range is an estimate only and the manufacturer make no guarantees that this range will be achieved for your specific application.

ORDERING INFORMATION

SKU	Description	Region
CTPCTZ-503	PCTZ-503 Smart Thermostat	All

RECOMMENDED ACCESSORIES

SKU	Description	Region
CT200M-508	CT200 Smart Gateway (CAT-M1 Modem)	All
CT200i-505	CT200 Smart Gateway (3G Modem)	All

Go to www.carbonTRACK.com.au for detailed product model numbers.

SERVICE & WARRANTY

carbonTRACK's comprehensive Support Service programs offer a full array of options to suit your specific needs. These services are aimed at protecting your investment and reducing the total cost of ownership.

TECHNICAL SUPPORT SERVICES

At carbonTRACK, we are committed to providing you with a quality service with a high attention to detail. Several options of support are available to choose from.

For additional information on Support Services as well as other service offerings, please contact your carbonTRACK representative or visit www.carbonTRACK.com.au

HEAD OFFICE
carbonTRACK Limited
Level 2
104 Burwood Road
Hawthorn, Victoria
Australia 3122
Tel: 1300 288 648
Email: info@carbontrack.com.au
www.carbonTRACK.com.au

APPENDIX F: CARBONTRACK CLIMATE COMMAND SPECIFICATIONS

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

Climate Command

Infrared Split System Controller

Control your Heating and Cooling

The Climate Command device is an all in one control module for split air conditioning systems, encapsulated in a small form factor and can be controlled and configured remotely via the carbonTRACK Smart Hub.

Flexibility

The Climate Command device is a very smart, adaptable control device.

It can be used with a number of leading split system air conditioning and heating systems currently available on the market.

BENEFITS

- Control and manage your split system temperature and running schedule
- Pre-installed IR codes for main stream split system air conditioners
- IR code learning functionality for unknown brand split system devices

FEATURES

- Control your device from the App & online dashboard
- Switch & smart control
- Device scheduling
- Reliable, real-time data
- ZigBee HA 1.2 compliant
- Safety & system alerts
- MDT data security

carbonTRACK®
INTELLIGENT ENERGY MANAGEMENT

SPECIFICATIONS

Model	CT-AC201
Radio Frequency ZigBee Power and Standard	2.4GHz Channels 11-26, +8dBm, ZigBee HA 1.2
Device Range (Outdoor / Indoor)	100 metres / 30 metres
Input Voltage and Current	100 – 240 VAC (50-60 Hz)
Rated Power Consumption	4-watts maximum
Firmware Upgrade Method	Not Supported
Mounting Type	Direct Plug in. US, EU, UK, AU
Physical Description	
Dimensions (W x L x H)	1.69” x 2.62” x 1.69” (43.00 mm x 66.50 mm x 43.00 mm)
Weight	0.26 lbs (0.116 kg)
Chassis Type	Plastic
Environmental	
Operating Temperature	-20° to +55° C*
Storage Temperature	-40° to +85° C
Relative Humidity	20% to 90%, non-condensing
Visual Characteristics	
Display	On screen display of room temperature
Certifications	
EMC Compliance	SANS 222 / CISPR 22, SANS / IEC 61000-3-3, SANS / IEC 61000-4-2, SANS / IEC 61000-4-3, SANS / IEC 61000-4-4, SANS / IEC 61000-4-5, SANS / IEC 61000-4-6, SANS / IEC 61000-4-8, SANS / IEC 61000-4-11
Radio Compliance	ACMA Section 376 of the Telecommunications Act 1997, AS/CA S042.1: 2011, AS/ACIF S042.3: 2005, AS/NZS 60950.1: 2011, ESTI EN 301 908-2 V5.4.1 (2012-12), ESTI EN 301 908-2 V6.2.1 (2013-04)
Safety Compliance	ANSI/UL 60950-1-2014, CSA/CAN C22.2 No. 60950-1-07, IEC60950-1 / SANS 60950-1, IEC61010-1
Environmental Compliance	Not Applicable

* Installation in outdoor locations or ambient temperature above 40° C or 70° C has not been evaluated by UL. UL Certification does not apply or extend to use in outdoor applications. Certification does not apply or extend to voltages outside certified range and has not been evaluated by UL for operating voltages beyond tested range.

GENERAL INFORMATION

Model	CT-AC201	
Functions	• Remote control of split system air conditioning and heating systems	
Schedules	• User configurable schedule-based control	
Data Transmission Protocol	• Secure carbonTRACK MDT protocol	
Monitoring		
Temperature	• Measuring range: -10° to +85° C	
Product Compatibility		
Supported Split System Units	• LG • Panasonic • Daiken • Toshiba • Fujitsu • Samsung • Sanyo • Mitsubishi • Hitachi • Sharp • Carrier • Whirlpool • Electrolux • Hisense • Haier • Midea • HAULING • WEILI • ChangLing • Inyan • RAYBO • AUX • TCL	• YuTu • DongBao • Conrowa • Aucma / SHENGFENG • YAIR • Panda • YORK • Sanken • JianRui • Wanbao • Polytron • SHANGLING • TIANYUAN • Galanz • ROWA • HUIFENG • Uchida • Huabao • Changhong • Gree • CHIGO • Kelon • Chunlan
Control Capabilities		
Infrared (IR)	• Infrared emission and receiving • Angle: 180° all-angle covering • Working temperature: 0° ~ 70° • Carrier frequency: 15kHz - 85kHz	

ORDERING INFORMATION

SKU	Description	Region
CTACCO-650	CT-CC carbonTRACK Split System Control Device	All

RECOMMENDED ACCESSORIES

SKU	Description	Region
CT200M-508	CT200 Smart Gateway (CAT-M1 Modem)	All
CT200i-505	CT200 Smart Gateway (3G Modem)	All
CTEYE-508	CT-EYE Monitor and Control Device	All
CTSMPL-750	Smart Plug	All

Go to www.carbonTRACK.com.au for detailed product model numbers.

SERVICE & WARRANTY

carbonTRACK's comprehensive Support Service programs offer a full array of options to suit your specific needs. These services are aimed at protecting your investment and reducing the total cost of ownership.

TECHNICAL SUPPORT SERVICES

At carbonTRACK, we are committed to providing you with a quality service with a high attention to detail. Several options of support are available to choose from.

For additional information on Support Services as well as other service offerings, please contact your carbonTRACK representative or visit www.carbonTRACK.com.au

HEAD OFFICE
carbonTRACK Limited
Level 2
104 Burwood Road
Hawthorn, Victoria
Australia 3122
Tel: 1300 288 648
Email: info@carbontrack.com.au
www.carbonTRACK.com.au

APPENDIX G: ESP32 SERIES DATASHEET

ESP32 Series

Datasheet

Including:

ESP32-D0WD-V3

ESP32-D0WDQ6-V3

ESP32-D0WD

ESP32-D0WDQ6

ESP32-D2WD

ESP32-S0WD

ESP32-U4WDH

Version 3.4

Espressif Systems

About This Guide

This document provides the specifications of ESP32 family of chips.

Document Updates

Please always refer to the latest version on https://www.espressif.com/en/support/download/documents.

Revision History

For any changes to this document over time, please refer to the last page.

Documentation Change Notification

Espressif provides email notifications to keep customers updated on changes to technical documentation. Please subscribe at www.espressif.com/en/subscribe. Note that you need to update your subscription to receive notifications of new products you are not currently subscribed to.

Certification

Download certificates for Espressif products from www.espressif.com/en/certificates.

Disclaimer and Copyright Notice

Contents

List of Tables

List of Figures

1. Overview

ESP32 is a single 2.4 GHz Wi-Fi-and-Bluetooth combo chip designed with the TSMC ultra-low-power 40 nm technology. It is designed to achieve the best power and RF performance, showing robustness, versatility and reliability in a wide variety of applications and power scenarios.

The ESP32 series of chips includes ESP32-D0WD-V3, ESP32-D0WDQ6-V3, ESP32-D0WD, ESP32-D0WDQ6, ESP32-D2WD, ESP32-S0WD, and ESP32-U4WDH, among which, ESP32-D0WD-V3, ESP32-D0WDQ6-V3, and ESP32-U4WDH are based on ECO V3 wafer.

For details on part numbers and ordering information, please refer to Section 7.

For details on ECO V3 instructions, please refer to *ESP32 ECO V3 User Guide*.

1.1 Featured Solutions

1.1.1 Ultra-Low-Power Solution

ESP32 is designed for mobile, wearable electronics, and Internet-of-Things (IoT) applications. It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock gating, multiple power modes, and dynamic power scaling. For instance, in a low-power IoT sensor hub application scenario, ESP32 is woken up periodically and only when a specified condition is detected. Low-duty cycle is used to minimize the amount of energy that the chip expends. The output of the power amplifier is also adjustable, thus contributing to an optimal trade-off between communication range, data rate and power consumption.

Note:
For more information, refer to Section 3.7 *RTC and Low-Power Management*.

1.1.2 Complete Integration Solution

ESP32 is a highly-integrated solution for Wi-Fi-and-Bluetooth IoT applications, with around 20 external components. ESP32 integrates an antenna switch, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules. As such, the entire solution occupies minimal Printed Circuit Board (PCB) area.

ESP32 uses CMOS for single-chip fully-integrated radio and baseband, while also integrating advanced calibration circuitries that allow the solution to remove external circuit imperfections or adjust to changes in external conditions. As such, the mass production of ESP32 solutions does not require expensive and specialized Wi-Fi testing equipment.

1.2 Wi-Fi Key Features

- 802.11 b/g/n
- 802.11 n (2.4 GHz), up to 150 Mbps
- WMM
- TX/RX A-MPDU, RX A-MSDU

- Immediate Block ACK
- Defragmentation
- Automatic Beacon monitoring (hardware TSF)
- 4 × virtual Wi-Fi interfaces
- Simultaneous support for Infrastructure Station, SoftAP, and Promiscuous modes
  Note that when ESP32 is in Station mode, performing a scan, the SoftAP channel will be changed.
- Antenna diversity

Note:
For more information, please refer to Section 3.5 *Wi-Fi*.

1.3 BT Key Features

- Compliant with Bluetooth v4.2 BR/EDR and BLE specifications
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced Power Control
- +12 dBm transmitting power
- NZIF receiver with –94 dBm BLE sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High-speed UART HCI, up to 4 Mbps
- Bluetooth 4.2 BR/EDR BLE dual mode controller
- Synchronous Connection-Oriented/Extended (SCO/eSCO)
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet
- Multi-connections in Classic BT and BLE
- Simultaneous advertising and scanning

1.4 MCU and Advanced Features

1.4.1 CPU and Memory

- Xtensa® single-/dual-core 32-bit LX6 microprocessor(s), up to 600 MIPS (200 MIPS for ESP32-S0WD/ESP32-U4WDH, 400 MIPS for ESP32-D2WD)
- 448 KB ROM
- 520 KB SRAM
- 16 KB SRAM in RTC
- QSPI supports multiple flash/SRAM chips

1.4.2 Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/BT functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including 2 × 64-bit timers and 1 × main watchdog in each group
- One RTC timer
- RTC watchdog

1.4.3 Advanced Peripheral Interfaces

- 34 × programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I²S
- 2 × I²C
- 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- CAN 2.0
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor

1.4.4 Security

- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration:
  - AES
  - Hash (SHA-2)
  - RSA

  - ECC
  - Random Number Generator (RNG)

1.5 Applications (A Non-exhaustive List)

- Generic Low-power IoT Sensor Hub
- Generic Low-power IoT Data Loggers
- Cameras for Video Streaming
- Over-the-top (OTT) Devices
- Speech Recognition
- Image Recognition
- Mesh Network
- Home Automation
  - Light control
  - Smart plugs
  - Smart door locks
- Smart Building
  - Smart lighting
  - Energy monitoring
- Industrial Automation
  - Industrial wireless control
  - Industrial robotics
- Smart Agriculture
  - Smart greenhouses
  - Smart irrigation
  - Agriculture robotics
- Audio Applications
  - Internet music players
  - Live streaming devices
  - Internet radio players
  - Audio headsets
- Health Care Applications
  - Health monitoring
  - Baby monitors
- Wi-Fi-enabled Toys
  - Remote control toys
  - Proximity sensing toys
  - Educational toys
- Wearable Electronics
  - Smart watches
  - Smart bracelets
- Retail & Catering Applications
  - POS machines
  - Service robots

1.6 Block Diagram

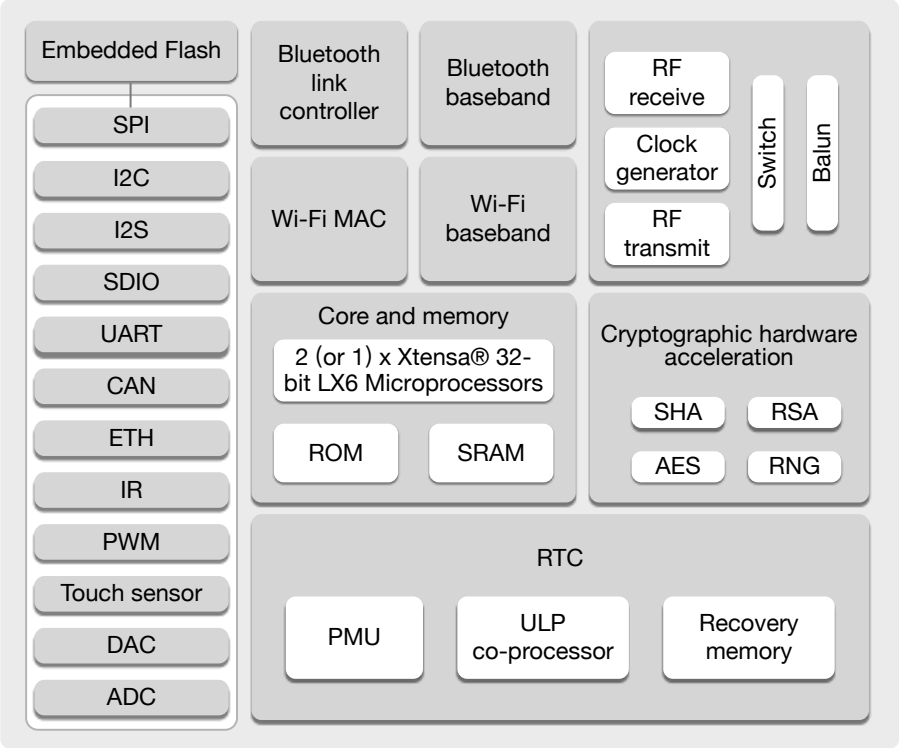


Figure 1: Functional Block Diagram

Note:
Products in the ESP32 series differ from each other in terms of their support for embedded flash and the number of CPUs they have. For details, please refer to Section 7 *Part Number and Ordering Information*.

2. Pin Definitions

2.1 Pin Layout

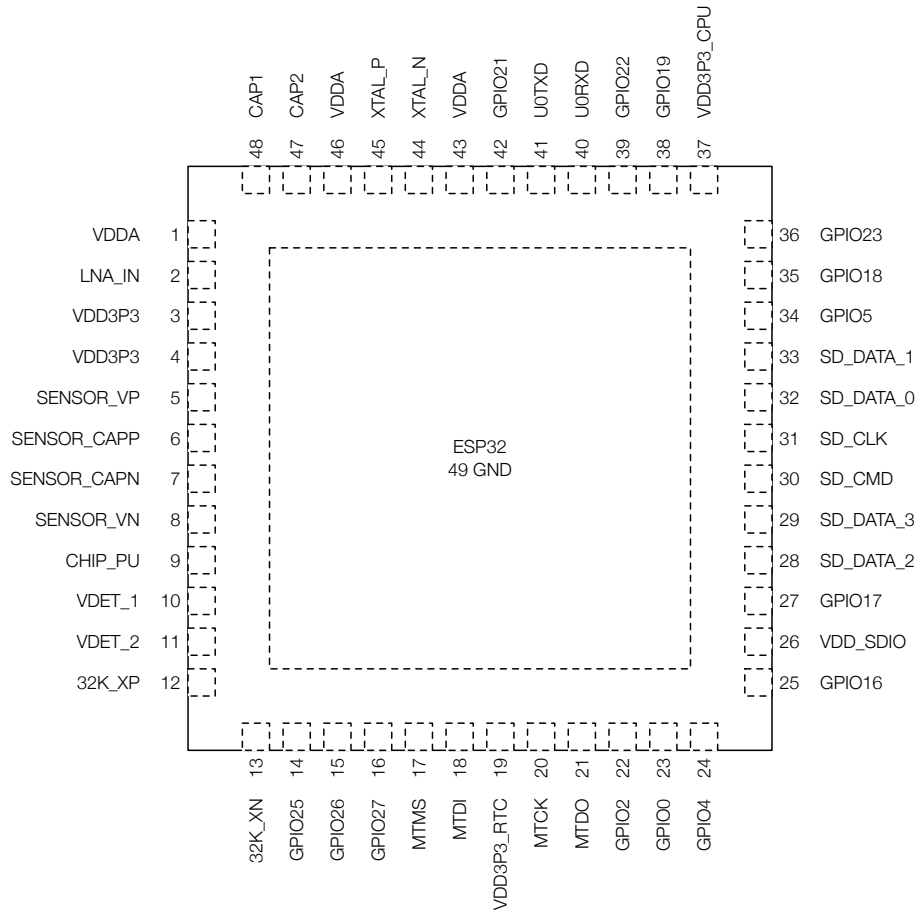


Figure 2: ESP32 Pin Layout (QFN 6*6, Top View)

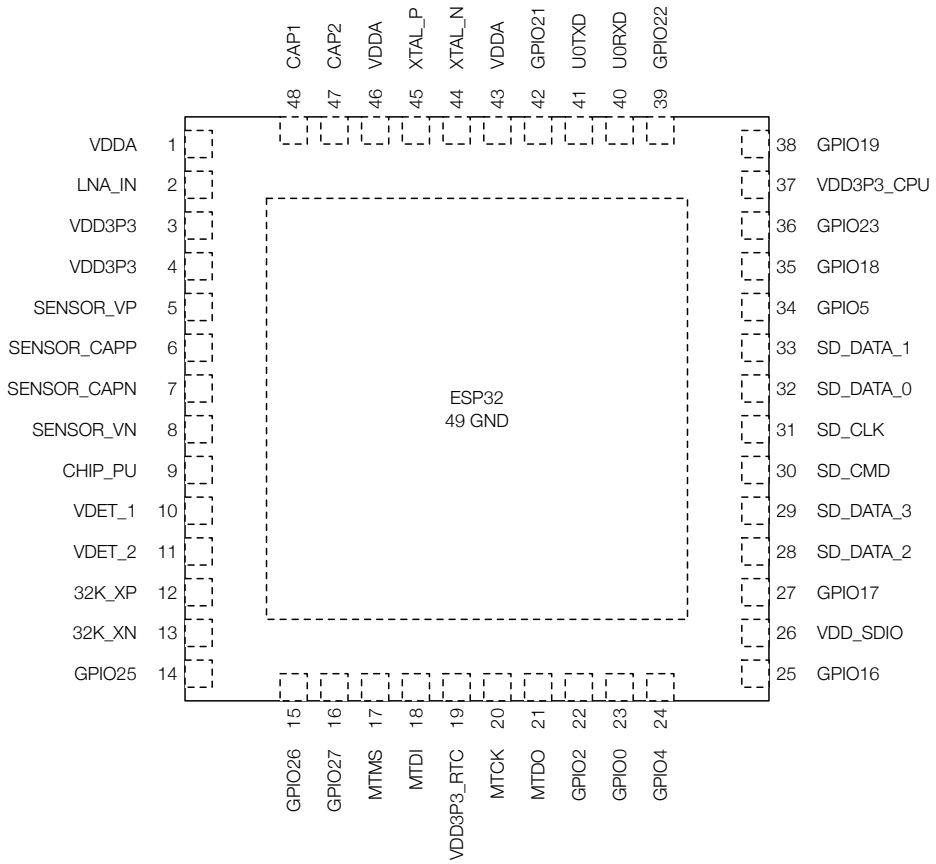


Figure 3: ESP32 Pin Layout (QFN 5*5, Top View)

Note:

For details on ESP32's part numbers and the corresponding packaging, please refer to Section 7 *Part Number and Ordering Information*.

2.2 Pin Description

Table 1: Pin Description

Name	No.	Type	Function
Analog			
VDDA	1	P	Analog power supply (2.3 V ~ 3.6 V)
LNA_IN	2	I/O	RF input and output
VDD3P3	3	P	Analog power supply (2.3 V ~ 3.6 V)
VDD3P3	4	P	Analog power supply (2.3 V ~ 3.6 V)
VDD3P3_RTC			
SENSOR_VP	5	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_CAPP	6	I	GPIO37, ADC1_CH1, RTC_GPIO1
SENSOR_CAPN	7	I	GPIO38, ADC1_CH2, RTC_GPIO2
SENSOR_VN	8	I	GPIO39, ADC1_CH3, RTC_GPIO3
CHIP_PU	9	I	High: On; enables the chip Low: Off; the chip powers off Note: Do not leave the CHIP_PU pin floating.
VDET_1	10	I	GPIO34, ADC1_CH6, RTC_GPIO4
VDET_2	11	I	GPIO35, ADC1_CH7, RTC_GPIO5
32K_XP	12	I/O	GPIO32, ADC1_CH4, RTC_GPIO9, TOUCH9, 32K_XP (32.768 kHz crystal oscillator input)
32K_XN	13	I/O	GPIO33, ADC1_CH5, RTC_GPIO8, TOUCH8, 32K_XN (32.768 kHz crystal oscillator output)
GPIO25	14	I/O	GPIO25, ADC2_CH8, RTC_GPIO6, DAC_1, EMAC_RXD0
GPIO26	15	I/O	GPIO26, ADC2_CH9, RTC_GPIO7, DAC_2, EMAC_RXD1
GPIO27	16	I/O	GPIO27, ADC2_CH7, RTC_GPIO17, TOUCH7, EMAC_RX_DV
MTMS	17	I/O	GPIO14, ADC2_CH6, RTC_GPIO16, TOUCH6, EMAC_TXD2, HSPICLK, HS2_CLK, SD_CLK, MTMS
MTDI	18	I/O	GPIO12, ADC2_CH5, RTC_GPIO15, TOUCH5, EMAC_TXD3, HSPIQ, HS2_DATA2, SD_DATA2, MTDI
VDD3P3_RTC	19	P	Input power supply for RTC IO (2.3 V ~ 3.6 V)
MTCK	20	I/O	GPIO13, ADC2_CH4, RTC_GPIO14, TOUCH4, EMAC_RX_ER, HSPID, HS2_DATA3, SD_DATA3, MTCK
MTDO	21	I/O	GPIO15, ADC2_CH3, RTC_GPIO13, TOUCH3, EMAC_RXD3, HSPICS0, HS2_CMD, SD_CMD, MTDO

Name	No.	Type	Function
GPIO2	22	I/O	GPIO2, ADC2_CH2, RTC_GPIO12, TOUCH2, HSPIWP, HS2_DATA0, SD_DATA0
GPIO0	23	I/O	GPIO0, ADC2_CH1, RTC_GPIO11, TOUCH1, EMAC_TX_CLK, CLK_OUT1,
GPIO4	24	I/O	GPIO4, ADC2_CH0, RTC_GPIO10, TOUCH0, EMAC_TX_ER, HSPIHD, HS2_DATA1, SD_DATA1
VDD_SDIO			
GPIO16	25	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
VDD_SDIO	26	P	Output power supply: 1.8 V or the same voltage as VDD3P3_RTC
GPIO17	27	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
SD_DATA_2	28	I/O	GPIO9, HS1_DATA2, U1RXD, SD_DATA2, SPIHD
SD_DATA_3	29	I/O	GPIO10, HS1_DATA3, U1TXD, SD_DATA3, SPIWP
SD_CMD	30	I/O	GPIO11, HS1_CMD, U1RTS, SD_CMD, SPICS0
SD_CLK	31	I/O	GPIO6, HS1_CLK, U1CTS, SD_CLK, SPICLK
SD_DATA_0	32	I/O	GPIO7, HS1_DATA0, U2RTS, SD_DATA0, SPIQ
SD_DATA_1	33	I/O	GPIO8, HS1_DATA1, U2CTS, SD_DATA1, SPID
VDD3P3_CPU			
GPIO5	34	I/O	GPIO5, HS1_DATA6, VSPICS0, EMAC_RX_CLK
GPIO18	35	I/O	GPIO18, HS1_DATA7, VSPICLK
GPIO23	36	I/O	GPIO23, HS1_STROBE, VSPID
VDD3P3_CPU	37	P	Input power supply for CPU IO (1.8 V ~ 3.6 V)
GPIO19	38	I/O	GPIO19, U0CTS, VSPIQ, EMAC_TXD0
GPIO22	39	I/O	GPIO22, U0RTS, VSPIWP, EMAC_TXD1
U0RXD	40	I/O	GPIO3, U0RXD, CLK_OUT2
U0TXD	41	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
GPIO21	42	I/O	GPIO21, VSPIHD, EMAC_TX_EN
Analog			
VDDA	43	P	Analog power supply (2.3 V ~ 3.6 V)
XTAL_N	44	O	External crystal output
XTAL_P	45	I	External crystal input
VDDA	46	P	Analog power supply (2.3 V ~ 3.6 V)
CAP2	47	I	Connects to a 3 nF capacitor and 20 kΩ resistor in parallel to CAP1

Name	No.	Type	Function
CAP1	48	I	Connects to a 10 nF series capacitor to ground
GND	49	P	Ground

Note:

- The pin-pin mapping between ESP32-D2WD/ESP32-U4WDH and the embedded flash is as follows: GPIO16 = CS#, GPIO17 = IO1/DO, SD_CMD = IO3/HOLD#, SD_CLK = CLK, SD_DATA_0 = IO2/WP#, SD_DATA_1 = IO0/DI. The pins used for embedded flash are not recommended for other uses.
- In most cases, the data port connection between ESP32 series of chips other than ESP32-D2WD/ESP32-U4WDH and external flash is as follows: SD_DATA0/SPIQ = IO1/DO, SD_DATA1/SPID = IO0/DI, SD_DATA2/SPIHD = IO3/HOLD#, SD_DATA3/SPIWP = IO2/WP#.
- For a quick reference guide to using the IO_MUX, Ethernet MAC, and GIPO Matrix pins of ESP32, please refer to Appendix *ESP32 Pin Lists*.

2.3 Power Scheme

ESP32's digital pins are divided into three different power domains:

- VDD3P3_RTC
- VDD3P3_CPU
- VDD_SDIO

VDD3P3_RTC is also the input power supply for RTC and CPU.

VDD3P3_CPU is also the input power supply for CPU.

VDD_SDIO connects to the output of an internal LDO whose input is VDD3P3_RTC. When VDD_SDIO is connected to the same PCB net together with VDD3P3_RTC, the internal LDO is disabled automatically. The power scheme diagram is shown below:

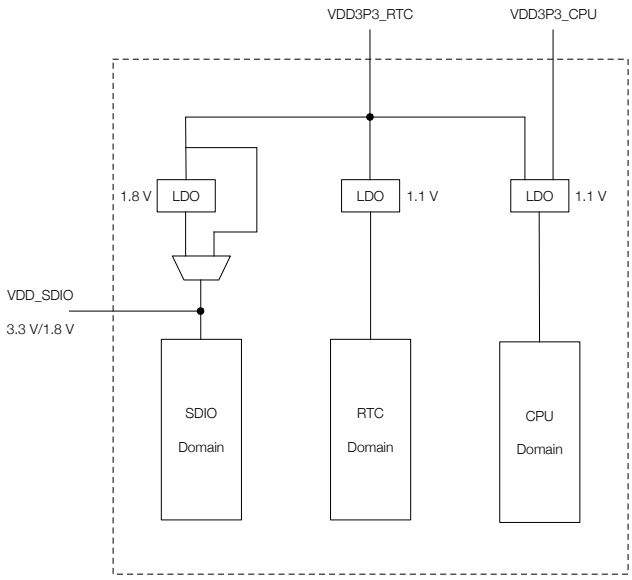


Figure 4: ESP32 Power Scheme

The internal LDO can be configured as having 1.8 V, or the same voltage as VDD3P3_RTC. It can be powered off via software to minimize the current of flash/SRAM during the Deep-sleep mode.

Notes on CHIP_PU:

- The illustration below shows the ESP32 power-up and reset timing. Details about the parameters are listed in Table 2.

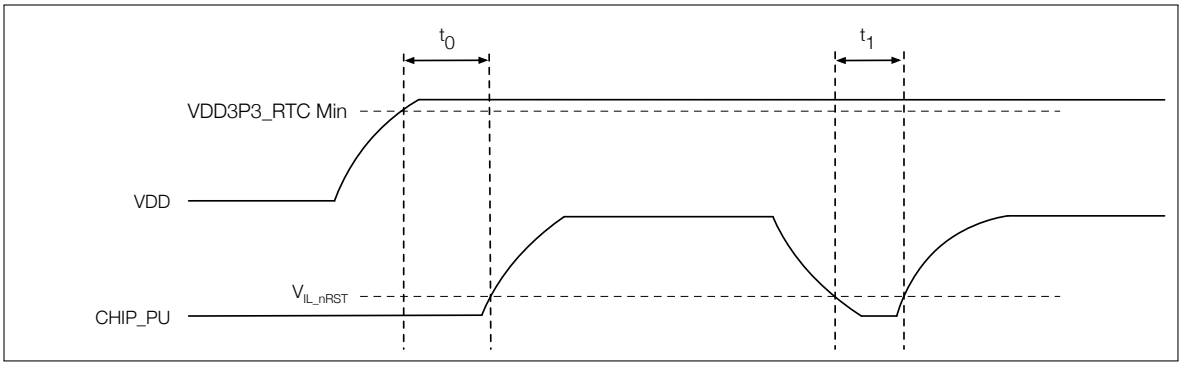


Figure 5: ESP32 Power-up and Reset Timing

Table 2: Description of ESP32 Power-up and Reset Timing Parameters

Parameters	Description	Min.	Unit
t_0	Time between the 3.3 V rails being brought up and CHIP_PU being activated	50	μs
t_1	Duration of CHIP_PU signal level < V_{IL_nRST} (refer to its value in Table 13 DC Characteristics) to reset the chip	50	μs

- In scenarios where ESP32 is powered on and off repeatedly by switching the power rails, while there is a large capacitor on the VDD33 rail and CHIP_PU and VDD33 are connected, simply switching off the CHIP_PU power rail and immediately switching it back on may cause an incomplete power discharge cycle and failure to reset the chip adequately.
  An additional discharge circuit may be required to accelerate the discharge of the large capacitor on rail VDD33, which will ensure proper power-on-reset when the ESP32 is powered up again. Please find the discharge circuit in Figure **ESP32-WROOM-32 Peripheral Schematics**, in *ESP32-WROOM-32 Datasheet*.
- When a battery is used as the power supply for the ESP32 series of chips and modules, a supply voltage supervisor is recommended, so that a boot failure due to low voltage is avoided. Users are recommended to pull CHIP_PU low if the power supply for ESP32 is below 2.3 V. For the reset circuit, please refer to Figure **ESP32-WROOM-32 Peripheral Schematics**, in *ESP32-WROOM-32 Datasheet*.

Notes on power supply:

- The operating voltage of ESP32 ranges from 2.3 V to 3.6 V. When using a single-power supply, the recommended voltage of the power supply is 3.3 V, and its recommended output current is 500 mA or more.
- When VDD_SDIO 1.8 V is used as the power supply for external flash/PSRAM, a 2-kohm grounding resistor should be added to VDD_SDIO. For the circuit design, please refer to Figure **ESP32-WROVER Schematics**, in *ESP32-WROVER Datasheet*.
- When the three digital power supplies are used to drive peripherals, e.g., 3.3 V flash, they should comply with the peripherals' specifications.

2.4 Strapping Pins

There are five strapping pins:

- MTDI
- GPIO0
- GPIO2
- MTDO
- GPIO5

Software can read the values of these five bits from register "GPIO_STRAPPING".

During the chip's system reset release (power-on-reset, RTC watchdog reset and brownout reset), the latches of the strapping pins sample the voltage level as strapping bits of "0" or "1", and hold these bits until the chip is powered down or shut down. The strapping bits configure the device's boot mode, the operating voltage of VDD_SDIO and other initial system settings.

Each strapping pin is connected to its internal pull-up/pull-down during the chip reset. Consequently, if a strapping pin is unconnected or the connected external circuit is high-impedance, the internal weak pull-up/pull-down will determine the default input level of the strapping pins.

To change the strapping bit values, users can apply the external pull-down/pull-up resistances, or use the host MCU's GPIOs to control the voltage level of these pins when powering on the chip.

After reset release, the strapping pins work as normal-function pins.

Refer to Table 3 for a detailed boot-mode configuration by strapping pins.

Table 3: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)					
Pin	Default	3.3 V		1.8 V	
MTDI	Pull-down	0		1	
Booting Mode					
Pin	Default	SPI Boot		Download Boot	
GPIO0	Pull-up	1		0	
GPIO2	Pull-down	Don't-care		0	
Enabling/Disabling Debugging Log Print over U0TXD During Booting					
Pin	Default	U0TXD Active		U0TXD Silent	
MTDO	Pull-up	1		0	
Timing of SDIO Slave					
Pin	Default	FE Sampling FE Output	FE Sampling RE Output	RE Sampling FE Output	RE Sampling RE Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

Note:

- FE: falling-edge, RE: rising-edge.
- Firmware can configure register bits to change the settings of "Voltage of Internal LDO (VDD_SDIO)" and "Timing of SDIO Slave", after booting.
- For ESP32 chips that contain an embedded flash, users need to note the logic level of MTDI. For example, ESP32-D2WD contains an embedded flash that operates at 1.8 V, therefore, the MTDI should be pulled high. ESP32-U4WDH contains an embedded flash that operates at 3.3 V, therefore, the MTDI should be low.

The illustration below shows the setup and hold times for the strapping pin before and after the CHIP_PU signal goes high. Details about the parameters are listed in Table 4.

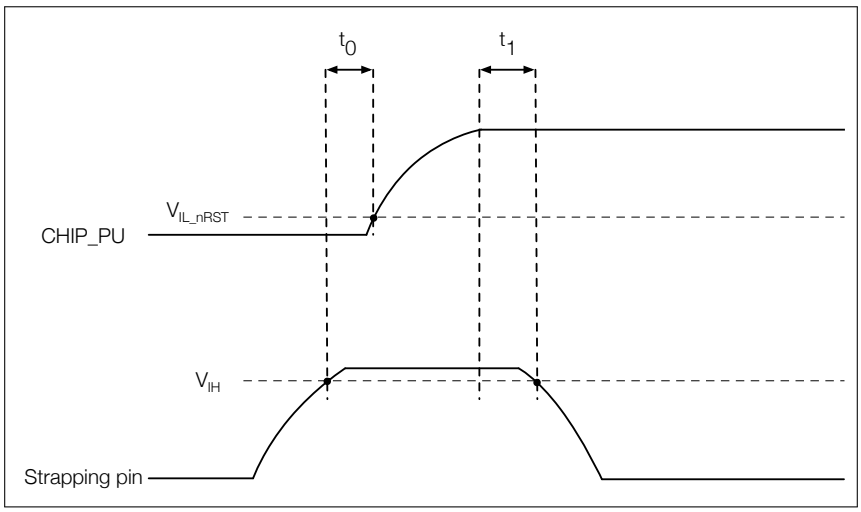


Figure 6: Setup and Hold Times for the Strapping Pin

Table 4: Parameter Descriptions of Setup and Hold Times for the Strapping Pin

Parameters	Description	Min.	Unit
t_0	Setup time before CHIP_PU goes from low to high	0	ms
t_1	Hold time after CHIP_PU goes high	1	ms

3. Functional Description

This chapter describes the functions integrated in ESP32.

3.1 CPU and Memory

3.1.1 CPU

ESP32 contains one or two low-power Xtensa® 32-bit LX6 microprocessor(s) with the following features:

- 7-stage pipeline to support the clock frequency of up to 240 MHz (160 MHz for ESP32-S0WD, ESP32-D2WD, and ESP32-U4WDH)
- 16/24-bit Instruction Set provides high code-density
- Support for Floating Point Unit
- Support for DSP instructions, such as a 32-bit multiplier, a 32-bit divider, and a 40-bit MAC
- Support for 32 interrupt vectors from about 70 interrupt sources

The single-/dual-CPU interfaces include:

- Xtensa RAM/ROM Interface for instructions and data
- Xtensa Local Memory Interface for fast peripheral register access
- External and internal interrupt sources
- JTAG for debugging

3.1.2 Internal Memory

ESP32's internal memory includes:

- 448 KB of ROM for booting and core functions
- 520 KB of on-chip SRAM for data and instructions
- 8 KB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.
- 8 KB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the co-processor during the Deep-sleep mode.
- 1 Kbit of eFuse: 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including flash-encryption and chip-ID.
- Embedded flash

Note:
Products in the ESP32 series differ from each other, in terms of their support for embedded flash and the size of it. For details, please refer to Section 7 *Part Number and Ordering Information*.

3.1.3 External Flash and SRAM

ESP32 supports multiple external QSPI flash and SRAM chips. More details can be found in Chapter SPI in the *ESP32 Technical Reference Manual*. ESP32 also supports hardware encryption/decryption based on AES to protect developers' programs and data in flash.

ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- Up to 16 MB of external flash can be mapped into CPU instruction memory space and read-only memory space simultaneously.
  - When external flash is mapped into CPU instruction memory space, up to 11 MB + 248 KB can be mapped at a time. Note that if more than 3 MB + 248 KB are mapped, cache performance will be reduced due to speculative reads by the CPU.
  - When external flash is mapped into read-only data memory space, up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads are supported.
- External SRAM can be mapped into CPU data memory space. SRAM up to 8 MB is supported and up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads and writes are supported.

Note:
After ESP32 is initialized, firmware can customize the mapping of external SRAM or flash into the CPU address space.

3.1.4 Memory Map

The structure of address mapping is shown in Figure 7. The memory and peripheral mapping of ESP32 is shown in Table 5.

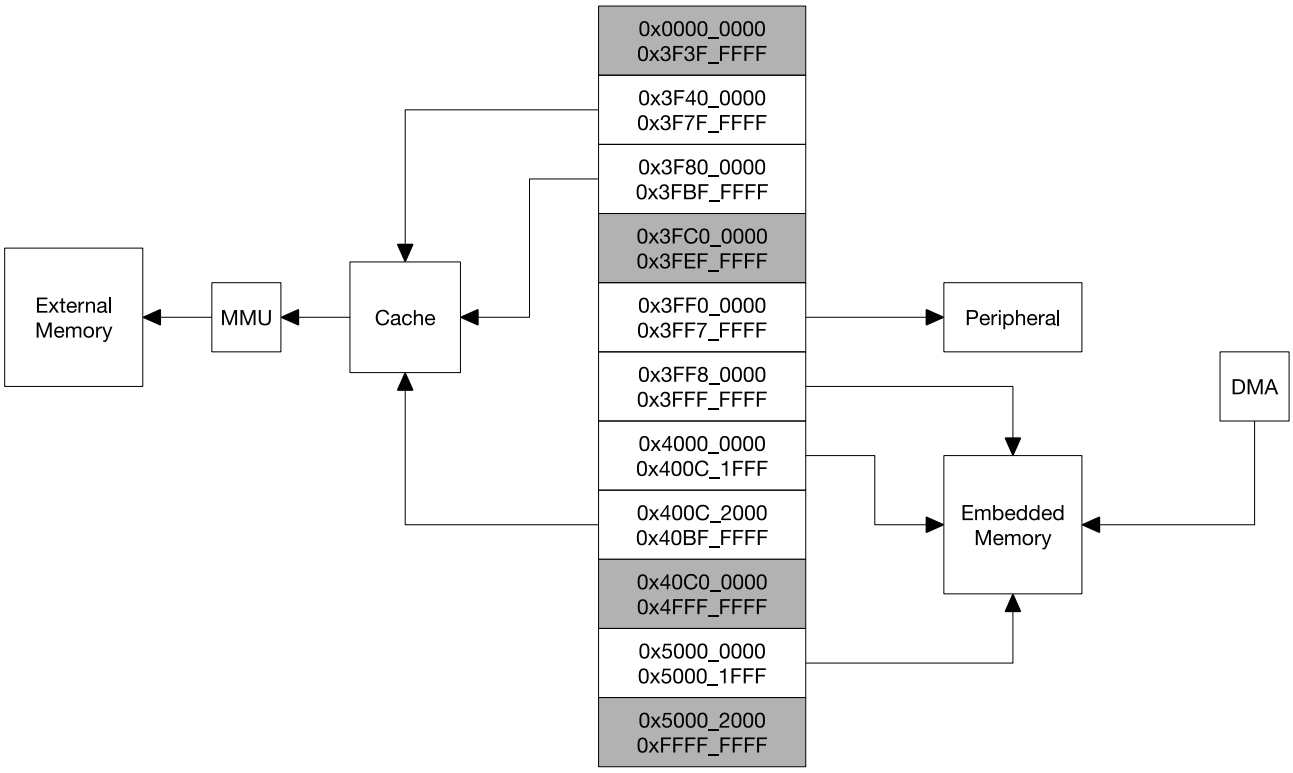


Figure 7: Address Mapping Structure

Table 5: Memory and Peripheral Mapping

Category	Target	Start Address	End Address	Size
Embedded Memory	Internal ROM 0	0x4000_0000	0x4005_FFFF	384 KB
	Internal ROM 1	0x3FF9_0000	0x3FF9_FFFF	64 KB
	Internal SRAM 0	0x4007_0000	0x4009_FFFF	192 KB
	Internal SRAM 1	0x3FFE_0000	0x3FFF_FFFF	128 KB
		0x400A_0000	0x400B_FFFF	
	Internal SRAM 2	0x3FFA_E000	0x3FFD_FFFF	200 KB
	RTC FAST Memory	0x3FF8_0000	0x3FF8_1FFF	8 KB
		0x400C_0000	0x400C_1FFF	
	RTC SLOW Memory	0x5000_0000	0x5000_1FFF	8 KB
External Memory	External Flash	0x3F40_0000	0x3F7F_FFFF	4 MB
		0x400C_2000	0x40BF_FFFF	11 MB+248 KB
	External RAM	0x3F80_0000	0x3FBF_FFFF	4 MB
Peripheral	DPort Register	0x3FF0_0000	0x3FF0_0FFF	4 KB
	AES Accelerator	0x3FF0_1000	0x3FF0_1FFF	4 KB
	RSA Accelerator	0x3FF0_2000	0x3FF0_2FFF	4 KB
	SHA Accelerator	0x3FF0_3000	0x3FF0_3FFF	4 KB
	Secure Boot	0x3FF0_4000	0x3FF0_4FFF	4 KB
	Cache MMU Table	0x3FF1_0000	0x3FF1_3FFF	16 KB
	PID Controller	0x3FF1_F000	0x3FF1_FFFF	4 KB
	UART0	0x3FF4_0000	0x3FF4_0FFF	4 KB
	SPI1	0x3FF4_2000	0x3FF4_2FFF	4 KB
	SPI0	0x3FF4_3000	0x3FF4_3FFF	4 KB
	GPIO	0x3FF4_4000	0x3FF4_4FFF	4 KB
	RTC	0x3FF4_8000	0x3FF4_8FFF	4 KB
	IO MUX	0x3FF4_9000	0x3FF4_9FFF	4 KB
	SDIO Slave	0x3FF4_B000	0x3FF4_BFFF	4 KB
	UDMA1	0x3FF4_C000	0x3FF4_CFFF	4 KB
	I2S0	0x3FF4_F000	0x3FF4_FFFF	4 KB
	UART1	0x3FF5_0000	0x3FF5_0FFF	4 KB
	I2C0	0x3FF5_3000	0x3FF5_3FFF	4 KB
	UDMA0	0x3FF5_4000	0x3FF5_4FFF	4 KB
	SDIO Slave	0x3FF5_5000	0x3FF5_5FFF	4 KB
	RMT	0x3FF5_6000	0x3FF5_6FFF	4 KB
	PCNT	0x3FF5_7000	0x3FF5_7FFF	4 KB
	SDIO Slave	0x3FF5_8000	0x3FF5_8FFF	4 KB
	LED PWM	0x3FF5_9000	0x3FF5_9FFF	4 KB
	Efuse Controller	0x3FF5_A000	0x3FF5_AFFF	4 KB
	Flash Encryption	0x3FF5_B000	0x3FF5_BFFF	4 KB
	PWM0	0x3FF5_E000	0x3FF5_EFFF	4 KB
	TIMG0	0x3FF5_F000	0x3FF5_FFFF	4 KB
	TIMG1	0x3FF6_0000	0x3FF6_0FFF	4 KB

Category	Target	Start Address	End Address	Size
Peripheral	SPI2	0x3FF6_4000	0x3FF6_4FFF	4 KB
	SPI3	0x3FF6_5000	0x3FF6_5FFF	4 KB
	SYSCON	0x3FF6_6000	0x3FF6_6FFF	4 KB
	I2C1	0x3FF6_7000	0x3FF6_7FFF	4 KB
	SDMMC	0x3FF6_8000	0x3FF6_8FFF	4 KB
	EMAC	0x3FF6_9000	0x3FF6_AFFF	8 KB
	PWM1	0x3FF6_C000	0x3FF6_CFFF	4 KB
	I2S1	0x3FF6_D000	0x3FF6_DFFF	4 KB
	UART2	0x3FF6_E000	0x3FF6_EFFF	4 KB
	PWM2	0x3FF6_F000	0x3FF6_FFFF	4 KB
	PWM3	0x3FF7_0000	0x3FF7_0FFF	4 KB
	RNG	0x3FF7_5000	0x3FF7_5FFF	4 KB

3.2 Timers and Watchdogs

3.2.1 64-bit Timers

There are four general-purpose timers embedded in the chip. They are all 64-bit generic timers which are based on 16-bit prescalers and 64-bit auto-reload-capable up/down-timers.

The timers feature:

- A 16-bit clock prescaler, from 2 to 65536
- A 64-bit timer
- Configurable up/down timer: incrementing or decrementing
- Halt and resume of time-base counter
- Auto-reload at alarming
- Software-controlled instant reload
- Level and edge interrupt generation

3.2.2 Watchdog Timers

The chip has three watchdog timers: one in each of the two timer modules (called the Main Watchdog Timer, or MWDT) and one in the RTC module (called the RTC Watchdog Timer, or RWDT). These watchdog timers are intended to recover from an unforeseen fault causing the application program to abandon its normal sequence. A watchdog timer has four stages. Each stage may trigger one of three or four possible actions upon the expiry of its programmed time period, unless the watchdog is fed or disabled. The actions are: interrupt, CPU reset, core reset, and system reset. Only the RWDT can trigger the system reset, and is able to reset the entire chip, including the RTC itself. A timeout value can be set for each stage individually.

During flash boot the RWDT and the first MWDT start automatically in order to detect, and recover from, booting problems.

The watchdogs have the following features:

- Four stages, each of which can be configured or disabled separately

- A programmable time period for each stage
- One of three or four possible actions (interrupt, CPU reset, core reset, and system reset) upon the expiry of each stage
- 32-bit expiry counter
- Write protection that prevents the RWDT and MWDT configuration from being inadvertently altered
- SPI flash boot protection
  If the boot process from an SPI flash does not complete within a predetermined time period, the watchdog will reboot the entire system.

3.3 System Clocks

3.3.1 CPU Clock

Upon reset, an external crystal clock source is selected as the default CPU clock. The external crystal clock source also connects to a PLL to generate a high-frequency clock (typically 160 MHz).

In addition, ESP32 has an internal 8 MHz oscillator. The application can select the clock source from the external crystal clock source, the PLL clock or the internal 8 MHz oscillator. The selected clock source drives the CPU clock directly, or after division, depending on the application.

3.3.2 RTC Clock

The RTC clock has five possible sources:

- external low-speed (32 kHz) crystal clock
- external crystal clock divided by 4
- internal RC oscillator (typically about 150 kHz, and adjustable)
- internal 8 MHz oscillator
- internal 31.25 kHz clock (derived from the internal 8 MHz oscillator divided by 256)

When the chip is in the normal power mode and needs faster CPU accessing, the application can choose the external high-speed crystal clock divided by 4 or the internal 8 MHz oscillator. When the chip operates in the low-power mode, the application chooses the external low-speed (32 kHz) crystal clock, the internal RC clock or the internal 31.25 kHz clock.

3.3.3 Audio PLL Clock

The audio clock is generated by the ultra-low-noise fractional-N PLL. More details can be found in Chapter Reset and Clock in the *ESP32 Technical Reference Manual*.

3.4 Radio

The radio module consists of the following blocks:

- 2.4 GHz receiver
- 2.4 GHz transmitter

- bias and regulators
- balun and transmit-receive switch
- clock generator

3.4.1 2.4 GHz Receiver

The 2.4 GHz receiver demodulates the 2.4 GHz RF signal to quadrature baseband signals and converts them to the digital domain with two high-resolution, high-speed ADCs. To adapt to varying signal channel conditions, RF filters, Automatic Gain Control (AGC), DC offset cancelation circuits and baseband filters are integrated in the chip.

3.4.2 2.4 GHz Transmitter

The 2.4 GHz transmitter modulates the quadrature baseband signals to the 2.4 GHz RF signal, and drives the antenna with a high-powered Complementary Metal Oxide Semiconductor (CMOS) power amplifier. The use of digital calibration further improves the linearity of the power amplifier, enabling state-of-the-art performance in delivering up to +20.5 dBm of power for an 802.11b transmission and +18 dBm for an 802.11n transmission.

Additional calibrations are integrated to cancel any radio imperfections, such as:

- Carrier leakage
- I/Q phase matching
- Baseband nonlinearities
- RF nonlinearities
- Antenna matching

These built-in calibration routines reduce the amount of time required for product testing, and render the testing equipment unnecessary.

3.4.3 Clock Generator

The clock generator produces quadrature clock signals of 2.4 GHz for both the receiver and the transmitter. All components of the clock generator are integrated into the chip, including all inductors, varactors, filters, regulators and dividers.

The clock generator has built-in calibration and self-test circuits. Quadrature clock phases and phase noise are optimized on-chip with patented calibration algorithms which ensure the best performance of the receiver and the transmitter.

3.5 Wi-Fi

ESP32 implements a TCP/IP and full 802.11 b/g/n Wi-Fi MAC protocol. It supports the Basic Service Set (BSS) STA and SoftAP operations under the Distributed Control Function (DCF). Power management is handled with minimal host interaction to minimize the active-duty period.

3.5.1 Wi-Fi Radio and Baseband

The ESP32 Wi-Fi Radio and Baseband support the following features:

- 802.11b/g/n
- 802.11n MCS0-7 in both 20 MHz and 40 MHz bandwidth
- 802.11n MCS32 (RX)
- 802.11n 0.4 μs guard-interval
- up to 150 Mbps of data rate
- Receiving STBC 2×1
- Up to 20.5 dBm of transmitting power
- Adjustable transmitting power
- Antenna diversity
  ESP32 supports antenna diversity with an external RF switch. One or more GPIOs control the RF switch and selects the best antenna to minimize the effects of channel fading.

3.5.2 Wi-Fi MAC

The ESP32 Wi-Fi MAC applies low-level protocol functions automatically. They are as follows:

- 4 × virtual Wi-Fi interfaces
- Simultaneous Infrastructure BSS Station mode/SoftAP mode/Promiscuous mode
- RTS protection, CTS protection, Immediate Block ACK
- Defragmentation
- TX/RX A-MPDU, RX A-MSDU
- TXOP
- WMM
- CCMP (CBC-MAC, counter mode), TKIP (MIC, RC4), WAPI (SMS4), WEP (RC4) and CRC
- Automatic beacon monitoring (hardware TSF)

3.6 Bluetooth

The chip integrates a Bluetooth link controller and Bluetooth baseband, which carry out the baseband protocols and other low-level link routines, such as modulation/demodulation, packet processing, bit stream processing, frequency hopping, etc.

3.6.1 Bluetooth Radio and Baseband

The Bluetooth Radio and Baseband support the following features:

- Class-1, class-2 and class-3 transmit output powers, and a dynamic control range of up to 24 dB
- π/4 DQPSK and 8 DPSK modulation
- High performance in NZIF receiver sensitivity with over 94 dBm of dynamic range

- Class-1 operation without external PA
- Internal SRAM allows full-speed data-transfer, mixed voice and data, and full piconet operation
- Logic for forward error correction, header error control, access code correlation, CRC, demodulation, encryption bit stream generation, whitening and transmit pulse shaping
- ACL, SCO, eSCO and AFH
- A-law, μ-law and CVSD digital audio CODEC in PCM interface
- SBC audio CODEC
- Power management for low-power applications
- SMP with 128-bit AES

3.6.2 Bluetooth Interface

- Provides UART HCI interface, up to 4 Mbps
- Provides SDIO / SPI HCI interface
- Provides PCM / I²S audio interface

3.6.3 Bluetooth Stack

The Bluetooth stack of the chip is compliant with the Bluetooth v4.2 BR/EDR and Bluetooth LE specifications.

3.6.4 Bluetooth Link Controller

The link controller operates in three major states: standby, connection and sniff. It enables multiple connections, and other operations, such as inquiry, page, and secure simple-pairing, and therefore enables Piconet and Scatternet. Below are the features:

- Classic Bluetooth
  - Device Discovery (inquiry, and inquiry scan)
  - Connection establishment (page, and page scan)
  - Multi-connections
  - Asynchronous data reception and transmission
  - Synchronous links (SCO/eSCO)
  - Master/Slave Switch
  - Adaptive Frequency Hopping and Channel assessment
  - Broadcast encryption
  - Authentication and encryption
  - Secure Simple-Pairing
  - Multi-point and scatternet management
  - Sniff mode
  - Connectionless Slave Broadcast (transmitter and receiver)

  - Enhanced power control
  - Ping
- Bluetooth Low Energy
  - Advertising
  - Scanning
  - Simultaneous advertising and scanning
  - Multiple connections
  - Asynchronous data reception and transmission
  - Adaptive Frequency Hopping and Channel assessment
  - Connection parameter update
  - Data Length Extension
  - Link Layer Encryption
  - LE Ping

3.7 RTC and Low-Power Management

With the use of advanced power-management technologies, ESP32 can switch between different power modes.

- Power modes
  - **Active mode**: The chip radio is powered on. The chip can receive, transmit, or listen.
  - **Modem-sleep mode**: The CPU is operational and the clock is configurable. The Wi-Fi/Bluetooth baseband and radio are disabled.
  - **Light-sleep mode**: The CPU is paused. The RTC memory and RTC peripherals, as well as the ULP co-processor are running. Any wake-up events (MAC, host, RTC timer, or external interrupts) will wake up the chip.
  - **Deep-sleep mode**: Only the RTC memory and RTC peripherals are powered on. Wi-Fi and Bluetooth connection data are stored in the RTC memory. The ULP co-processor is functional.
  - **Hibernation mode**: The internal 8-MHz oscillator and ULP co-processor are disabled. The RTC recovery memory is powered down. Only one RTC timer on the slow clock and certain RTC GPIOs are active. The RTC timer or the RTC GPIOs can wake up the chip from the Hibernation mode.

Table 6: Power Consumption by Power Modes

Power mode	Description			Power consumption
Active (RF working)	Wi-Fi Tx packet			Please refer to Table 15 for details.
	Wi-Fi/BT Tx packet			
	Wi-Fi/BT Rx and listening			
Modem-sleep	The CPU is powered on.	240 MHz *	Dual-core chip(s)	30 mA ~ 68 mA
			Single-core chip(s)	N/A
		160 MHz *	Dual-core chip(s)	27 mA ~ 44 mA
			Single-core chip(s)	27 mA ~ 34 mA
		Normal speed: 80 MHz	Dual-core chip(s)	20 mA ~ 31 mA

Power mode	Description			Power consumption
			Single-core chip(s)	20 mA ~ 25 mA
Light-sleep	-			0.8 mA
Deep-sleep	The ULP co-processor is powered on.			150 μA
	ULP sensor-monitored pattern			100 μA @1% duty
	RTC timer + RTC memory			10 μA
Hibernation	RTC timer only			5 μA
Power off	CHIP_PU is set to low level, the chip is powered off.			1 μA

Note:

- * Among the ESP32 series of SoCs, ESP32-D0WD-V3, ESP32-D0WDQ6-V3, ESP32-D0WD, and ESP32-D0WDQ6 have a maximum CPU frequency of 240 MHz, ESP32-D2WD, ESP32-S0WD, and ESP32-U4WDH have a maximum CPU frequency of 160 MHz.
- When Wi-Fi is enabled, the chip switches between Active and Modem-sleep modes. Therefore, power consumption changes accordingly.
- In Modem-sleep mode, the CPU frequency changes automatically. The frequency depends on the CPU load and the peripherals used.
- During Deep-sleep, when the ULP co-processor is powered on, peripherals such as GPIO and I^2C are able to operate.
- When the system works in the ULP sensor-monitored pattern, the ULP co-processor works with the ULP sensor periodically and the ADC works with a duty cycle of 1%, so the power consumption is 100 μA.

4. Peripherals and Sensors

4.1 Descriptions of Peripherals and Sensors

4.1.1 General Purpose Input / Output Interface (GPIO)

ESP32 has 34 GPIO pins which can be assigned various functions by programming the appropriate registers. There are several kinds of GPIOs: digital-only, analog-enabled, capacitive-touch-enabled, etc. Analog-enabled GPIOs and Capacitive-touch-enabled GPIOs can be configured as digital GPIOs.

Most of the digital GPIOs can be configured as internal pull-up or pull-down, or set to high impedance. When configured as an input, the input value can be read through the register. The input can also be set to edge-trigger or level-trigger to generate CPU interrupts. Most of the digital IO pins are bi-directional, non-inverting and tristate, including input and output buffers with tristate control. These pins can be multiplexed with other functions, such as the SDIO, UART, SPI, etc. (More details can be found in the Appendix, Table IO_MUX.) For low-power operations, the GPIOs can be set to hold their states.

4.1.2 Analog-to-Digital Converter (ADC)

ESP32 integrates 12-bit SAR ADCs and supports measurements on 18 channels (analog-enabled pins). The ULP-coprocessor in ESP32 is also designed to measure voltage, while operating in the sleep mode, which enables low-power consumption. The CPU can be woken up by a threshold setting and/or via other triggers.

With appropriate settings, the ADCs can be configured to measure voltage on 18 pins maximum.

Table 7 describes the ADC characteristics.

Table 7: ADC Characteristics

Parameter	Description	Min	Max	Unit
DNL (Differential nonlinearity)	RTC controller; ADC connected to an external 100 nF capacitor; DC signal input; ambient temperature at 25 °C; Wi-Fi&BT off	–7	7	LSB
INL (Integral nonlinearity)		–12	12	LSB
Sampling rate	RTC controller	-	200	ksps
	DIG controller	-	2	Msps

Notes:

- When atten=3 and the measurement result is above 3000 (voltage at approx. 2450 mV), the ADC accuracy will be worse than described in the table above.
- To get better DNL results, users can take multiple sampling tests with a filter, or calculate the average value.
- The input voltage range of GPIO pins within VDD3P3_RTC domain should strictly follow the DC characteristics provided in Table 13. Otherwise, measurement errors may be introduced, and chip performance may be affected.

By default, there are ±6% differences in measured results between chips. ESP-IDF provides couple of calibration methods for ADC1. Results after calibration using eFuse Vref value are shown in Table 8. For higher accuracy, users may apply other calibration methods provided in ESP-IDF, or implement their own.

Table 8: ADC Calibration Results

Parameter	Description	Min	Max	Unit
Total error	Atten=0, effective measurement range of 100 ~ 950 mV	–23	23	mV
	Atten=1, effective measurement range of 100 ~ 1250 mV	–30	30	mV
	Atten=2, effective measurement range of 150 ~ 1750 mV	–40	40	mV
	Atten=3, effective measurement range of 150 ~ 2450 mV	–60	60	mV

4.1.3 Hall Sensor

ESP32 integrates a Hall sensor based on an N-carrier resistor. When the chip is in the magnetic field, the Hall sensor develops a small voltage laterally on the resistor, which can be directly measured by the ADC.

4.1.4 Digital-to-Analog Converter (DAC)

Two 8-bit DAC channels can be used to convert two digital signals into two analog voltage signal outputs. The design structure is composed of integrated resistor strings and a buffer. This dual DAC supports power supply as input voltage reference. The two DAC channels can also support independent conversions.

4.1.5 Touch Sensor

ESP32 has 10 capacitive-sensing GPIOs, which detect variations induced by touching or approaching the GPIOs with a finger or other objects. The low-noise nature of the design and the high sensitivity of the circuit allow relatively small pads to be used. Arrays of pads can also be used, so that a larger area or more points can be detected. The 10 capacitive-sensing GPIOs are listed in Table 9.

Table 9: Capacitive-Sensing GPIOs Available on ESP32

Capacitive-sensing signal name	Pin name
T0	GPIO4
T1	GPIO0
T2	GPIO2
T3	MTDO
T4	MTCK
T5	MTDI
T6	MTMS
T7	GPIO27
T8	32K_XN
T9	32K_XP

4.1.6 Ultra-Low-Power Co-processor

The ULP processor and RTC memory remain powered on during the Deep-sleep mode. Hence, the developer can store a program for the ULP processor in the RTC slow memory to access the peripheral devices, internal timers and internal sensors during the Deep-sleep mode. This is useful for designing applications where the CPU needs to be woken up by an external event, or a timer, or a combination of the two, while maintaining minimal power consumption.

4.1.7 Ethernet MAC Interface

An IEEE-802.3-2008-compliant Media Access Controller (MAC) is provided for Ethernet LAN communications. ESP32 requires an external physical interface device (PHY) to connect to the physical LAN bus (twisted-pair, fiber, etc.). The PHY is connected to ESP32 through 17 signals of MII or nine signals of RMII. The following features are supported on the Ethernet MAC (EMAC) interface:

- 10 Mbps and 100 Mbps rates
- Dedicated DMA controller allowing high-speed transfer between the dedicated SRAM and Ethernet MAC
- Tagged MAC frame (VLAN support)
- Half-duplex (CSMA/CD) and full-duplex operation
- MAC control sublayer (control frames)
- 32-bit CRC generation and removal
- Several address-filtering modes for physical and multicast address (multicast and group addresses)
- 32-bit status code for each transmitted or received frame
- Internal FIFOs to buffer transmit and receive frames. The transmit FIFO and the receive FIFO are both 512 words (32-bit)
- Hardware PTP (Precision Time Protocol) in accordance with IEEE 1588 2008 (PTP V2)
- 25 MHz/50 MHz clock output

4.1.8 SD/SDIO/MMC Host Controller

An SD/SDIO/MMC host controller is available on ESP32, which supports the following features:

- Secure Digital memory (SD mem Version 3.0 and Version 3.01)
- Secure Digital I/O (SDIO Version 3.0)
- Consumer Electronics Advanced Transport Architecture (CE-ATA Version 1.1)
- Multimedia Cards (MMC Version 4.41, eMMC Version 4.5 and Version 4.51)

The controller allows up to 80 MHz clock output in three different data-bus modes: 1-bit, 4-bit and 8-bit. It supports two SD/SDIO/MMC4.41 cards in a 4-bit data-bus mode. It also supports one SD card operating at 1.8 V.

4.1.9 SDIO/SPI Slave Controller

ESP32 integrates an SD device interface that conforms to the industry-standard SDIO Card Specification Version 2.0, and allows a host controller to access the SoC, using the SDIO bus interface and protocol. ESP32 acts as the slave on the SDIO bus. The host can access the SDIO-interface registers directly and can access shared memory via a DMA engine, thus maximizing performance without engaging the processor cores.

The SDIO/SPI slave controller supports the following features:

- SPI, 1-bit SDIO, and 4-bit SDIO transfer modes over the full clock range from 0 to 50 MHz
- Configurable sampling and driving clock edge
- Special registers for direct access by host
- Interrupts to host for initiating data transfer

- Automatic loading of SDIO bus data and automatic discarding of padding data
- Block size of up to 512 bytes
- Interrupt vectors between the host and the slave, allowing both to interrupt each other
- Supports DMA for data transfer

4.1.10 Universal Asynchronous Receiver Transmitter (UART)

ESP32 has three UART interfaces, i.e., UART0, UART1 and UART2, which provide asynchronous communication (RS232 and RS485) and IrDA support, communicating at a speed of up to 5 Mbps. UART provides hardware management of the CTS and RTS signals and software flow control (XON and XOFF). All of the interfaces can be accessed by the DMA controller or directly by the CPU.

4.1.11 I²C Interface

ESP32 has two I²C bus interfaces which can serve as I²C master or slave, depending on the user's configuration. The I²C interfaces support:

- Standard mode (100 Kbit/s)
- Fast mode (400 Kbit/s)
- Up to 5 MHz, yet constrained by SDA pull-up strength
- 7-bit/10-bit addressing mode
- Dual addressing mode

Users can program command registers to control I²C interfaces, so that they have more flexibility.

4.1.12 I²S Interface

Two standard I²S interfaces are available in ESP32. They can be operated in master or slave mode, in full duplex and half-duplex communication modes, and can be configured to operate with an 8-/16-/32-/48-/64-bit resolution as input or output channels. BCK clock frequency, from 10 kHz up to 40 MHz, is supported. When one or both of the I²S interfaces are configured in the master mode, the master clock can be output to the external DAC/CODEC.

Both of the I²S interfaces have dedicated DMA controllers. PDM and BT PCM interfaces are supported.

4.1.13 Infrared Remote Controller

The infrared remote controller supports eight channels of infrared remote transmission and receiving. By programming the pulse waveform, it supports various infrared protocols. Eight channels share a 512 x 32-bit block of memory to store the transmitting or receiving waveform.

4.1.14 Pulse Counter

The pulse counter captures pulse and counts pulse edges through seven modes. It has eight channels, each of which captures four signals at a time. The four input signals include two pulse signals and two control signals. When the counter reaches a defined threshold, an interrupt is generated.

4.1.15 Pulse Width Modulation (PWM)

The Pulse Width Modulation (PWM) controller can be used for driving digital motors and smart lights. The controller consists of PWM timers, the PWM operator and a dedicated capture sub-module. Each timer provides timing in synchronous or independent form, and each PWM operator generates a waveform for one PWM channel. The dedicated capture sub-module can accurately capture events with external timing.

4.1.16 LED PWM

The LED PWM controller can generate 16 independent channels of digital waveforms with configurable periods and duties.

The 16 channels of digital waveforms operate with an APB clock of 80 MHz. Eight of these channels have the option of using the 8 MHz oscillator clock. Each channel can select a 20-bit timer with configurable counting range, while its accuracy of duty can be up to 16 bits within a 1 ms period.

The software can change the duty immediately. Moreover, each channel automatically supports step-by-step duty increase or decrease, which is useful for the LED RGB color-gradient generator.

4.1.17 Serial Peripheral Interface (SPI)

ESP32 features three SPIs (SPI, HSPI and VSPI) in slave and master modes in 1-line full-duplex and 1/2/4-line half-duplex communication modes. These SPIs also support the following general-purpose SPI features:

- Four modes of SPI transfer format, which depend on the polarity (CPOL) and the phase (CPHA) of the SPI clock
- Up to 80 MHz (The actual speed it can reach depends on the selected pads, PCB tracing, peripheral characteristics, etc.)
- up to 64-byte FIFO

All SPIs can also be connected to the external flash/SRAM and LCD. Each SPI can be served by DMA controllers.

4.1.18 Accelerator

ESP32 is equipped with hardware accelerators of general algorithms, such as AES (FIPS PUB 197), SHA (FIPS PUB 180-4), RSA, and ECC, which support independent arithmetic, such as Big Integer Multiplication and Big Integer Modular Multiplication. The maximum operation length for RSA, ECC, Big Integer Multiply and Big Integer Modular Multiplication is 4096 bits.

The hardware accelerators greatly improve operation speed and reduce software complexity. They also support code encryption and dynamic decryption, which ensures that code in the flash will not be hacked.

4.2 Peripheral Pin Configurations

Table 10: Peripheral Pin Configurations

Interface	Signal	Pin	Function
ADC	ADC1_CH0	SENSOR_VP	Two 12-bit SAR ADCs
	ADC1_CH1	SENSOR_CAPP	
	ADC1_CH2	SENSOR_CAPN	
	ADC1_CH3	SENSOR_VN	
	ADC1_CH4	32K_XP	
	ADC1_CH5	32K_XN	
	ADC1_CH6	VDET_1	
	ADC1_CH7	VDET_2	
	ADC2_CH0	GPIO4	
	ADC2_CH1	GPIO0	
	ADC2_CH2	GPIO2	
	ADC2_CH3	MTDO	
	ADC2_CH4	MTCK	
	ADC2_CH5	MTDI	
	ADC2_CH6	MTMS	
	ADC2_CH7	GPIO27	
	ADC2_CH8	GPIO25	
	ADC2_CH9	GPIO26	
DAC	DAC_1	GPIO25	Two 8-bit DACs
	DAC_2	GPIO26	
Touch Sensor	TOUCH0	GPIO4	Capacitive touch sensors
	TOUCH1	GPIO0	
	TOUCH2	GPIO2	
	TOUCH3	MTDO	
	TOUCH4	MTCK	
	TOUCH5	MTDI	
	TOUCH6	MTMS	
	TOUCH7	GPIO27	
	TOUCH8	32K_XN	
	TOUCH9	32K_XP	
JTAG	MTDI	MTDI	JTAG for software debugging
	MTCK	MTCK	
	MTMS	MTMS	
	MTDO	MTDO	

Interface	Signal	Pin	Function
SD/SDIO/MMC Host Controller	HS2_CLK	MTMS	Supports SD memory card V3.01 standard
	HS2_CMD	MTDO	
	HS2_DATA0	GPIO2	
	HS2_DATA1	GPIO4	
	HS2_DATA2	MTDI	
	HS2_DATA3	MTCK	
Motor PWM	PWM0_OUT0~2	Any GPIO Pins	Three channels of 16-bit timers generate PWM waveforms. Each channel has a pair of output signals, three fault detection signals, three event-capture signals, and three sync signals.
	PWM1_OUT_IN0~2		
	PWM0_FLT_IN0~2		
	PWM1_FLT_IN0~2		
	PWM0_CAP_IN0~2		
	PWM1_CAP_IN0~2		
	PWM0_SYNC_IN0~2		
	PWM1_SYNC_IN0~2		
SDIO/SPI Slave Controller	SD_CLK	MTMS	SDIO interface that conforms to the industry standard SDIO 2.0 card specification
	SD_CMD	MTDO	
	SD_DATA0	GPIO2	
	SD_DATA1	GPIO4	
	SD_DATA2	MTDI	
	SD_DATA3	MTCK	
UART	U0RXD_in	Any GPIO Pins	Two UART devices with hardware flow-control and DMA
	U0CTS_in		
	U0DSR_in		
	U0TXD_out		
	U0RTS_out		
	U0DTR_out		
	U1RXD_in		
	U1CTS_in		
	U1TXD_out		
	U1RTS_out		
	U2RXD_in		
	U2CTS_in		
	U2TXD_out		
	U2RTS_out		
I²C	I2CEXT0_SCL_in	Any GPIO Pins	Two I²C devices in slave or master mode
	I2CEXT0_SDA_in		
	I2CEXT1_SCL_in		
	I2CEXT1_SDA_in		
	I2CEXT0_SCL_out		
	I2CEXT0_SDA_out		
	I2CEXT1_SCL_out		
	I2CEXT1_SDA_out		

Interface	Signal	Pin	Function
LED PWM	ledc_hs_sig_out0~7	Any GPIO Pins	16 independent channels @80 MHz clock/RTC CLK. Duty accuracy: 16 bits.
	ledc_ls_sig_out0~7		
I2S	I2S0I_DATA_in0~15	Any GPIO Pins	Stereo input and output from/to the audio codec; parallel LCD data output; parallel camera data input
	I2S0O_BCK_in		
	I2S0O_WS_in		
	I2S0I_BCK_in		
	I2S0I_WS_in		
	I2S0I_H_SYNC		
	I2S0I_V_SYNC		
	I2S0I_H_ENABLE		
	I2S0O_BCK_out		
	I2S0O_WS_out		
	I2S0I_BCK_out		
	I2S0I_WS_out		
	I2S0O_DATA_out0~23		
	I2S1I_DATA_in0~15		
	I2S1O_BCK_in		
	I2S1O_WS_in		
	I2S1I_BCK_in		
	I2S1I_WS_in		
	I2S1I_H_SYNC		
	I2S1I_V_SYNC		
	I2S1I_H_ENABLE		
	I2S1O_BCK_out		
	I2S1O_WS_out		
	I2S1I_BCK_out		
	I2S1I_WS_out		
	I2S1O_DATA_out0~23		
Infrared Remote Controller	RMT_SIG_IN0~7	Any GPIO Pins	Eight channels for an IR transmitter and receiver of various waveforms
	RMT_SIG_OUT0~7		
General Purpose SPI	HSPIQ_in/_out	Any GPIO Pins	Standard SPI consists of clock, chip-select, MOSI and MISO. These SPIs can be connected to LCD and other external devices. They support the following features: • Both master and slave modes; • Four sub-modes of the SPI transfer format; • Configurable SPI frequency; • Up to 64 bytes of FIFO and DMA.
	HSPID_in/_out		
	HSPICLK_in/_out		
	HSPI_CS0_in/_out		
	HSPI_CS1_out		
	HSPI_CS2_out		
	VSPIQ_in/_out		
	VSPID_in/_out		
	VSPICLK_in/_out		
	VSPI_CS0_in/_out		
	VSPI_CS1_out		
	VSPI_CS2_out		

Interface	Signal	Pin	Function
Parallel QSPI	SPIHD	SD_DATA_2	Supports Standard SPI, Dual SPI, and Quad SPI that can be connected to the external flash and SRAM
	SPIWP	SD_DATA_3	
	SPICS0	SD_CMD	
	SPICLK	SD_CLK	
	SPIQ	SD_DATA_0	
	SPID	SD_DATA_1	
	HSPICLK	MTMS	
	HSPICS0	MTDO	
	HSPIQ	MTDI	
	HSPID	MTCK	
	HSPIHD	GPIO4	
	HSPIWP	GPIO2	
	VSPICLK	GPIO18	
	VSPICS0	GPIO5	
	VSPIQ	GPIO19	
	VSPID	GPIO23	
	VSPIHD	GPIO21	
	VSPIWP	GPIO22	
EMAC	EMAC_TX_CLK	GPIO0	Ethernet MAC with MII/RMII interface
	EMAC_RX_CLK	GPIO5	
	EMAC_TX_EN	GPIO21	
	EMAC_TXD0	GPIO19	
	EMAC_TXD1	GPIO22	
	EMAC_TXD2	MTMS	
	EMAC_TXD3	MTDI	
	EMAC_RX_ER	MTCK	
	EMAC_RX_DV	GPIO27	
	EMAC_RXD0	GPIO25	
	EMAC_RXD1	GPIO26	
	EMAC_RXD2	U0TXD	
	EMAC_RXD3	MTDO	
	EMAC_CLK_OUT	GPIO16	
	EMAC_CLK_OUT_180	GPIO17	
	EMAC_TX_ER	GPIO4	
	EMAC_MDC_out	Any GPIO Pins	
	EMAC_MDI_in	Any GPIO Pins	
	EMAC_MDO_out	Any GPIO Pins	
	EMAC_CRS_out	Any GPIO Pins	
	EMAC_COL_out	Any GPIO Pins	

Interface	Signal	Pin	Function
Pulse Counter	pcnt_sig_ch0_in0	Any GPIO Pins	Operating in seven different modes, the pulse counter captures pulse and counts pulse edges.
	pcnt_sig_ch1_in0		
	pcnt_ctrl_ch0_in0		
	pcnt_ctrl_ch1_in0		
	pcnt_sig_ch0_in1		
	pcnt_sig_ch1_in1		
	pcnt_ctrl_ch0_in1		
	pcnt_ctrl_ch1_in1		
	pcnt_sig_ch0_in2		
	pcnt_sig_ch1_in2		
	pcnt_ctrl_ch0_in2		
	pcnt_ctrl_ch1_in2		
	pcnt_sig_ch0_in3		
	pcnt_sig_ch1_in3		
	pcnt_ctrl_ch0_in3		
	pcnt_ctrl_ch1_in3		
	pcnt_sig_ch0_in4		
	pcnt_sig_ch1_in4		
	pcnt_ctrl_ch0_in4		
	pcnt_ctrl_ch1_in4		
	pcnt_sig_ch0_in5		
	pcnt_sig_ch1_in5		
	pcnt_ctrl_ch0_in5		
	pcnt_ctrl_ch1_in5		
	pcnt_sig_ch0_in6		
	pcnt_sig_ch1_in6		
	pcnt_ctrl_ch0_in6		
	pcnt_ctrl_ch1_in6		
	pcnt_sig_ch0_in7		
	pcnt_sig_ch1_in7		
	pcnt_ctrl_ch0_in7		
	pcnt_ctrl_ch1_in7		

5. Electrical Characteristics

5.1 Absolute Maximum Ratings

Stresses beyond the absolute maximum ratings listed in the table below may cause permanent damage to the device. These are stress ratings only, and do not refer to the functional operation of the device that should follow the recommended operating conditions.

Table 11: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
VDDA, VDD3P3, VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO	Voltage applied to power supply pins per power domain	–0.3	3.6	V
I_{output} *	Cumulative IO output current	-	1200	mA
T_{store}	Storage temperature	–40	150	°C

* The chip worked properly after a 24-hour test in ambient temperature at 25 °C, and the IOs in three domains (VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO) output high logic level to ground.

5.2 Recommended Operating Conditions

Table 12: Recommended Operating Conditions

Symbol	Parameter	Min	Typ	Max	Unit
VDDA, VDD3P3_RTC [1] VDD3P3, VDD_SDIO (3.3 V mode) [2]	Voltage applied to power supply pins per power domain	2.3	3.3	3.6	V
VDD3P3_CPU	Voltage applied to power supply pin	1.8	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	-	-	A
T [3]	Operating temperature	–40	-	125	°C

1. When writing eFuse, VDD3P3_RTC should be at least 3.3 V.
2. - VDD_SDIO works as the power supply for the related IO, and also for an external device. Please refer to the Appendix IO_MUX of this datasheet for more details.
   - VDD_SDIO can be sourced internally by the ESP32 from the VDD3P3_RTC power domain:
     - When VDD_SDIO operates at 3.3 V, it is driven directly by VDD3P3_RTC through a 6 Ω resistor, therefore, there will be some voltage drop from VDD3P3_RTC.
     - When VDD_SDIO operates at 1.8 V, it can be generated from ESP32's internal LDO. The maximum current this LDO can offer is 40 mA, and the output voltage range is 1.65 V ~ 2.0 V.
   - VDD_SDIO can also be driven by an external power supply.
   - Please refer to Power Scheme, section 2.3, for more information.
3. The operating temperature of ESP32-D2WD and ESP32-U4WDH ranges from –40 °C to 105 °C, due to the flash embedded in them. The other chips in this series have no embedded flash, so their range of operating temperatures is –40 °C ~ 125 °C.

5.3 DC Characteristics (3.3 V, 25 °C)

Table 13: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA
I_{OL}	Low-level sink current (VDD[1] = 3.3 V, V_{OL} = 0.495 V, output drive strength set to the maximum)		-	28	-	mA
R_{PU}	Resistance of internal pull-up resistor		-	45	-	kΩ
R_{PD}	Resistance of internal pull-down resistor		-	45	-	kΩ
V_{IL_nRST}	Low-level input voltage of CHIP_PU to power off the chip		-	-	0.6	V

Notes:

1. Please see Table IO_MUX for IO's power domain. VDD is the I/O voltage for a particular power domain of pins.
2. For VDD3P3_CPU and VDD3P3_RTC power domain, per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.
3. For VDD_SDIO power domain, per-pin current sourced in the same domain is gradually reduced from around 30 mA to around 10 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.

5.4 Reliability Qualifications

Table 14: Reliability Qualifications

Reliability tests	Standards	Test conditions	Result
Electro-Static Discharge Sensitivity (ESD), Charge Device Mode (CDM) [1]	JEDEC EIA/JESD22-C101	±500 V, all pins	Pass
Electro-Static Discharge Sensitivity (ESD), Human Body Mode (HBM) [2]	JEDEC EIA/JESD22-A114	±1500 V, all pins	Pass
Latch-up (Over-current test)	JEDEC STANDARD NO.78	±50 mA ~ ±200 mA, room temperature, test for IO	Pass
Latch-up (Over-voltage test)	JEDEC STANDARD NO.78	1.5 × Vmax, room temperature, test for V_{supply}	Pass

Reliability tests	Standards	Test conditions	Result
Moisture Sensitivity Level (MSL)	J-STD-020, MSL 3	30 °C, 60% RH, 192 hours, IR × 3 @260 °C	Pass

1. JEDEC document JEP157 states that 250 V CDM allows safe manufacturing with a standard ESD control process.
2. JEDEC document JEP155 states that 500 V HBM allows safe manufacturing with a standard ESD control process.

5.5 RF Power-Consumption Specifications

The power consumption measurements are taken with a 3.3 V supply at 25 °C of ambient temperature at the RF port. All transmitters' measurements are based on a 50% duty cycle.

Table 15: RF Power-Consumption Specifications

Mode	Min	Typ	Max	Unit
Transmit 802.11b, DSSS 1 Mbps, POUT = +19.5 dBm	-	240	-	mA
Transmit 802.11g, OFDM 54 Mbps, POUT = +16 dBm	-	190	-	mA
Transmit 802.11n, OFDM MCS7, POUT = +14 dBm	-	180	-	mA
Receive 802.11b/g/n	-	95 ~ 100	-	mA
Transmit BT/BLE, POUT = 0 dBm	-	130	-	mA
Receive BT/BLE	-	95 ~ 100	-	mA

5.6 Wi-Fi Radio

Table 16: Wi-Fi Radio Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Operating frequency range ^{note1}	-	2412	-	2484	MHz
Output impedance ^{note2}	-	-	*note 2*	-	Ω
TX power ^{note3}	11n, MCS7	12	13	14	dBm
	11b mode	18.5	19.5	20.5	dBm
Sensitivity	11b, 1 Mbps	-	–98	-	dBm
	11b, 11 Mbps	-	–88	-	dBm
	11g, 6 Mbps	-	–93	-	dBm
	11g, 54 Mbps	-	–75	-	dBm
	11n, HT20, MCS0	-	–93	-	dBm
	11n, HT20, MCS7	-	–73	-	dBm
	11n, HT40, MCS0	-	–90	-	dBm
	11n, HT40, MCS7	-	–70	-	dBm
Adjacent channel rejection	11g, 6 Mbps	-	27	-	dB
	11g, 54 Mbps	-	13	-	dB
	11n, HT20, MCS0	-	27	-	dB
	11n, HT20, MCS7	-	12	-	dB

1. Device should operate in the frequency range allocated by regional regulatory authorities. Target operating frequency range is configurable by software.
2. The typical value of ESP32's Wi-Fi radio output impedance is different between chips in different QFN packages. For ESP32 chips with a QFN 6×6 package, the value is 30+j10 Ω. For ESP32 chips with a QFN 5×5 package, the value is 35+j10 Ω.
3. Target TX power is configurable based on device or certification requirements.

5.7 Bluetooth Radio

5.7.1 Receiver – Basic Data Rate

Table 17: Receiver Characteristics – Basic Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @0.1% BER	-	–90	–89	–88	dBm
Maximum received signal @0.1% BER	-	0	-	-	dBm
Co-channel C/I	-	-	+7	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	-	–6	dB
	F = F0 – 1 MHz	-	-	–6	dB
	F = F0 + 2 MHz	-	-	–25	dB
	F = F0 – 2 MHz	-	-	–33	dB
	F = F0 + 3 MHz	-	-	–25	dB
	F = F0 – 3 MHz	-	-	–45	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	–10	-	-	dBm
	2000 MHz ~ 2400 MHz	–27	-	-	dBm
	2500 MHz ~ 3000 MHz	–27	-	-	dBm
	3000 MHz ~ 12.5 GHz	–10	-	-	dBm
Intermodulation	-	–36	-	-	dBm

5.7.2 Transmitter – Basic Data Rate

Table 18: Transmitter Characteristics – Basic Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power (see note under Table 18)	-	-	0	-	dBm
Gain control step	-	-	3	-	dB
RF power control range	-	–12	-	+9	dBm
+20 dB bandwidth	-	-	0.9	-	MHz
Adjacent channel transmit power	F = F0 ± 2 MHz	-	–47	-	dBm
	F = F0 ± 3 MHz	-	–55	-	dBm
	F = F0 ± > 3 MHz	-	–60	-	dBm
$\Delta f1_{avg}$	-	-	-	155	kHz
$\Delta f2_{max}$	-	133.7	-	-	kHz
$\Delta f2_{avg}/\Delta f1_{avg}$	-	-	0.92	-	-
ICFT	-	-	–7	-	kHz

Parameter	Conditions	Min	Typ	Max	Unit
Drift rate	-	-	0.7	-	kHz/50 μs
Drift (DH1)	-	-	6	-	kHz
Drift (DH5)	-	-	6	-	kHz

Note:
There are a total of eight power levels from 0 to 7, and the transmit power ranges from –12 dBm to 9 dBm. When the power level rises by 1, the transmit power increases by 3 dB. Power level 4 is used by default and the corresponding transmit power is 0 dBm.

5.7.3 Receiver – Enhanced Data Rate

Table 19: Receiver Characteristics – Enhanced Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
π/4 DQPSK					
Sensitivity @0.01% BER	-	–90	–89	–88	dBm
Maximum received signal @0.01% BER	-	-	0	-	dBm
Co-channel C/I	-	-	11	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	–7	-	dB
	F = F0 – 1 MHz	-	–7	-	dB
	F = F0 + 2 MHz	-	–25	-	dB
	F = F0 – 2 MHz	-	–35	-	dB
	F = F0 + 3 MHz	-	–25	-	dB
	F = F0 – 3 MHz	-	–45	-	dB
8DPSK					
Sensitivity @0.01% BER	-	–84	–83	–82	dBm
Maximum received signal @0.01% BER	-	-	–5	-	dBm
C/I c-channel	-	-	18	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	2	-	dB
	F = F0 – 1 MHz	-	2	-	dB
	F = F0 + 2 MHz	-	–25	-	dB
	F = F0 – 2 MHz	-	–25	-	dB
	F = F0 + 3 MHz	-	–25	-	dB
	F = F0 – 3 MHz	-	–38	-	dB

5.7.4 Transmitter – Enhanced Data Rate

Table 20: Transmitter Characteristics – Enhanced Data Rate

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power (see note under Table 18)	-	-	0	-	dBm
Gain control step	-	-	3	-	dB
RF power control range	-	–12	-	+9	dBm
π/4 DQPSK max w0	-	-	–0.72	-	kHz

Parameter	Conditions	Min	Typ	Max	Unit
π/4 DQPSK max wi	-	-	–6	-	kHz
π/4 DQPSK max |wi + w0|	-	-	–7.42	-	kHz
8DPSK max w0	-	-	0.7	-	kHz
8DPSK max wi	-	-	–9.6	-	kHz
8DPSK max |wi + w0|	-	-	–10	-	kHz
π/4 DQPSK modulation accuracy	RMS DEVM	-	4.28	-	%
	99% DEVM	-	100	-	%
	Peak DEVM	-	13.3	-	%
8 DPSK modulation accuracy	RMS DEVM	-	5.8	-	%
	99% DEVM	-	100	-	%
	Peak DEVM	-	14	-	%
In-band spurious emissions	F = F0 ± 1 MHz	-	–46	-	dBm
	F = F0 ± 2 MHz	-	–40	-	dBm
	F = F0 ± 3 MHz	-	–46	-	dBm
	F = F0 +/– > 3 MHz	-	-	–53	dBm
EDR differential phase coding	-	-	100	-	%

5.8 Bluetooth LE Radio

5.8.1 Receiver

Table 21: Receiver Characteristics – BLE

Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @30.8% PER	-	–94	–93	–92	dBm
Maximum received signal @30.8% PER	-	0	-	-	dBm
Co-channel C/I	-	-	+10	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	–5	-	dB
	F = F0 – 1 MHz	-	–5	-	dB
	F = F0 + 2 MHz	-	–25	-	dB
	F = F0 – 2 MHz	-	–35	-	dB
	F = F0 + 3 MHz	-	–25	-	dB
	F = F0 – 3 MHz	-	–45	-	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	–10	-	-	dBm
	2000 MHz ~ 2400 MHz	–27	-	-	dBm
	2500 MHz ~ 3000 MHz	–27	-	-	dBm
	3000 MHz ~ 12.5 GHz	–10	-	-	dBm
Intermodulation	-	–36	-	-	dBm

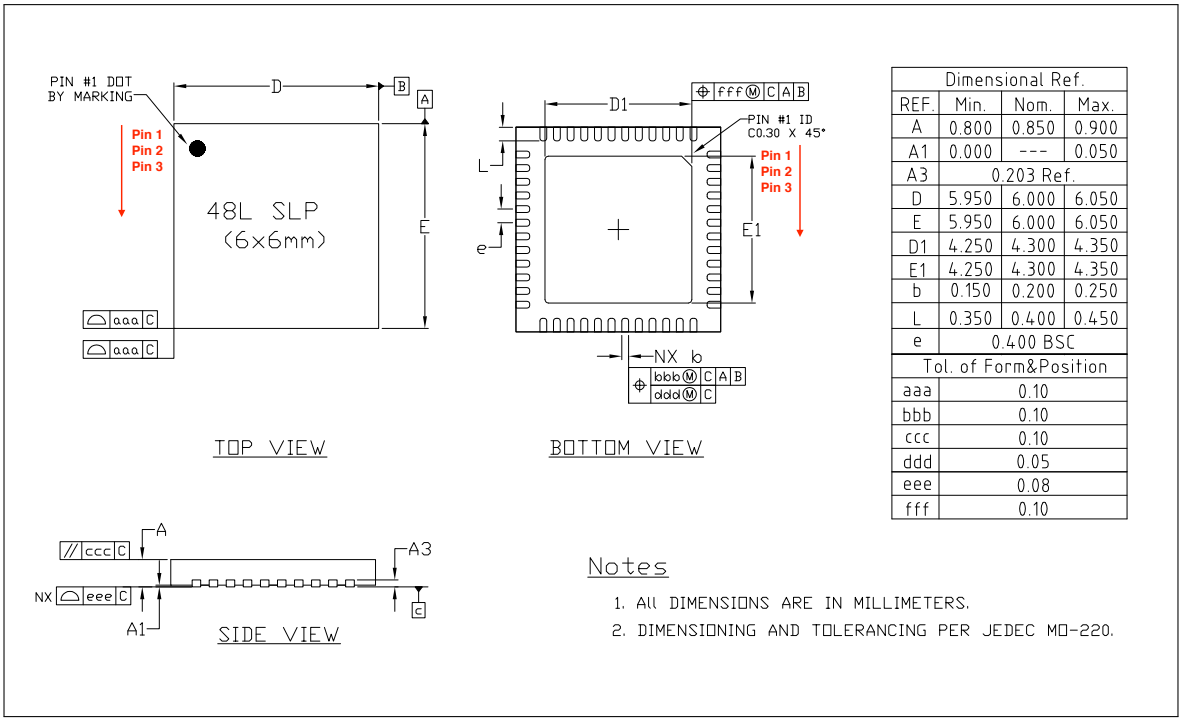
5.8.2 Transmitter

Table 22: Transmitter Characteristics – BLE

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power (see note under Table 18)	-	-	0	-	dBm

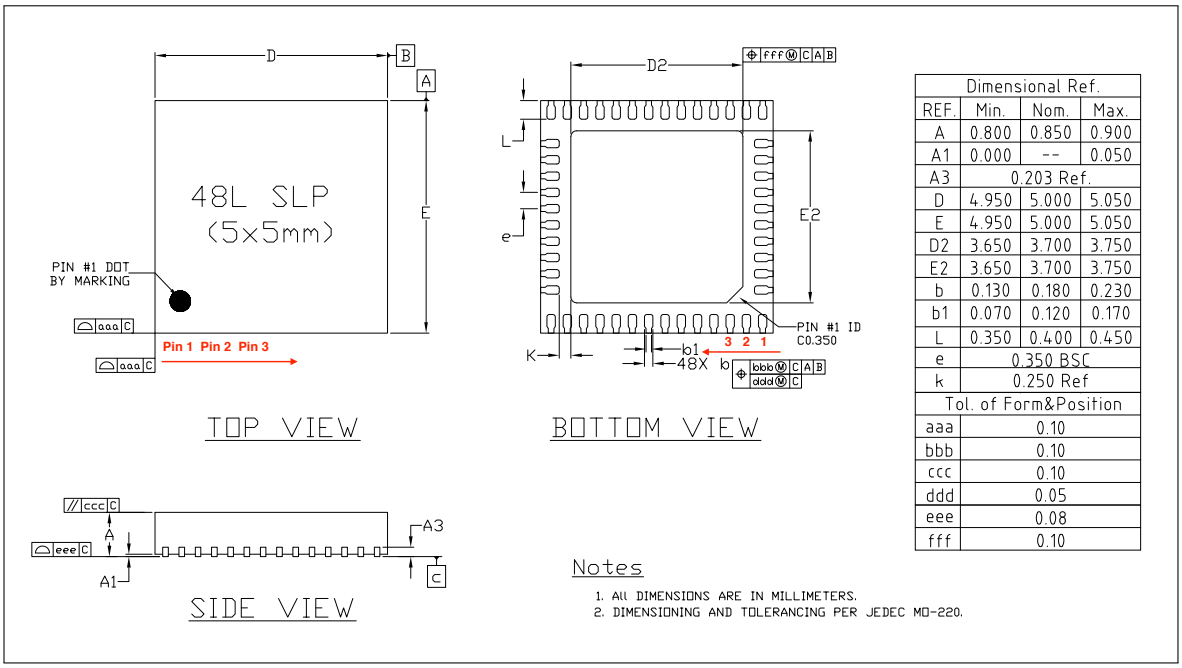
Parameter	Conditions	Min	Typ	Max	Unit
Gain control step	-	-	3	-	dB
RF power control range	-	–12	-	+9	dBm
Adjacent channel transmit power	F = F0 ± 2 MHz	-	–52	-	dBm
	F = F0 ± 3 MHz	-	–58	-	dBm
	F = F0 ± > 3 MHz	-	–60	-	dBm
$\Delta f1_{avg}$	-	-	-	265	kHz
$\Delta f2_{max}$	-	247	-	-	kHz
$\Delta f2_{avg}/\Delta f1_{avg}$	-	-	0.92	-	-
ICFT	-	-	–10	-	kHz
Drift rate	-	-	0.7	-	kHz/50 μs
Drift	-	-	2	-	kHz

6. Package Information



Dimensional Ref.			
REF.	Min.	Nom.	Max.
A	0.800	0.850	0.900
A1	0.000	---	0.050
A3	0.203 Ref.		
D	5.950	6.000	6.050
E	5.950	6.000	6.050
D1	4.250	4.300	4.350
E1	4.250	4.300	4.350
b	0.150	0.200	0.250
L	0.350	0.400	0.450
e	0.400 BSC		
Tol. of Form&Position			
aaa	0.10		
bbb	0.10		
ccc	0.10		
ddd	0.05		
eee	0.08		
fff	0.10		

Figure 8: QFN48 (6x6 mm) Package



Dimensional Ref.			
REF.	Min.	Nom.	Max.
A	0.800	0.850	0.900
A1	0.000	--	0.050
A3	0.203 Ref.		
D	4.950	5.000	5.050
E	4.950	5.000	5.050
D2	3.650	3.700	3.750
E2	3.650	3.700	3.750
b	0.130	0.180	0.230
b1	0.070	0.120	0.170
L	0.350	0.400	0.450
e	0.350 BSC		
k	0.250 Ref		
Tol. of Form&Position			
aaa	0.10		
bbb	0.10		
ccc	0.10		
ddd	0.05		
eee	0.08		
fff	0.10		

Figure 9: QFN48 (5x5 mm) Package

Note:
The pins of the chip are numbered in an anti-clockwise direction from Pin 1 in the top view.

7. Part Number and Ordering Information

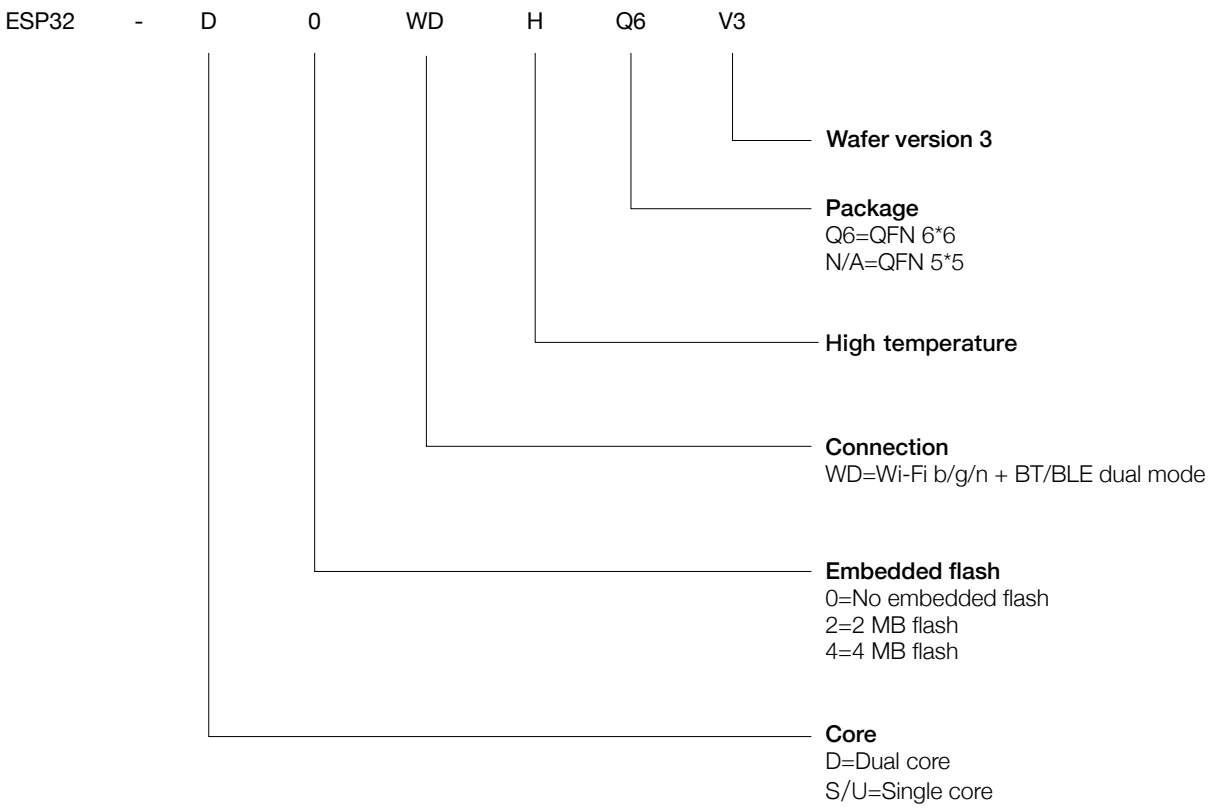


Figure 10: ESP32 Part Number

The table below provides the ordering information of the ESP32 series of chips.

Table 23: ESP32 Ordering Information

Ordering code	Core	Embedded flash	Package
ESP32-D0WD-V3	Dual core	No embedded flash	QFN 5*5
ESP32-D0WDQ6-V3	Dual core	No embedded flash	QFN 6*6
ESP32-D0WD	Dual core	No embedded flash	QFN 5*5
ESP32-D0WDQ6	Dual core	No embedded flash	QFN 6*6
ESP32-D2WD	Dual core	2 MB embedded flash (40 MHz)	QFN 5*5
ESP32-S0WD	Single core	No embedded flash	QFN 5*5
ESP32-U4WDH	Single core	4 MB embedded flash (80 MHz)	QFN 5*5
Note: All above chips support Wi-Fi b/g/n + BT/BLE Dual Mode connection.			

8. Learning Resources

8.1 Must-Read Documents

Click on the following links to access documents related to ESP32.

- *ESP32 ECO V3 User Guide*
  This document describes differences between V3 and previous ESP32 silicon wafer revisions.
- *ECO and Workarounds for Bugs in ESP32*
  This document details hardware errata and workarounds in the ESP32.
- *ESP-IDF Programming Guide*
  It hosts extensive documentation for ESP-IDF, ranging from hardware guides to API reference.
- *ESP32 Technical Reference Manual*
  The manual provides detailed information on how to use the ESP32 memory and peripherals.
- ESP32 Hardware Resources
  The zip files include schematics, PCB layout, Gerber and BOM list.
- *ESP32 Hardware Design Guidelines*
  The guidelines provide recommended design practices when developing standalone or add-on systems based on the ESP32 series of products, including the ESP32 chip, the ESP32 modules and development boards.
- *ESP32 AT Instruction Set and Examples*
  This document introduces the ESP32 AT commands, explains how to use them, and provides examples of several common AT commands.
- *Espressif Products Ordering Information*

8.2 Must-Have Resources

Here are the ESP32-related must-have resources.

- ESP32 BBS
  This is an Engineer-to-Engineer (E2E) Community for ESP32, where you can post questions, share knowledge, explore ideas, and solve problems together with fellow engineers.
- ESP32 GitHub
  ESP32 development projects are freely distributed under Espressif's MIT license on GitHub. This channel of communication has been established to help developers get started with ESP32 and encourage them to share their knowledge of ESP32-related hardware and software.
- ESP32 Tools
  This is a webpage where users can download ESP32 Flash Download Tools and the zip file "ESP32 Certification and Test".
- ESP-IDF
  This webpage links users to the official IoT development framework for ESP32.
- ESP32 Resources
  This webpage provides the links to all available ESP32 documents, SDK and tools.

Appendix A – ESP32 Pin Lists

A.1. Notes on ESP32 Pin Lists

Table 24: Notes on ESP32 Pin Lists

No.	Description
1	In Table IO_MUX, the boxes highlighted in yellow indicate the GPIO pins that are input-only. Please see the following note for further details.
2	GPIO pins 34-39 are input-only. These pins do not feature an output driver or internal pull-up/pull-down circuitry. The pin names are: SENSOR_VP (GPIO36), SENSOR_CAPP (GPIO37), SENSOR_CAPN (GPIO38), SENSOR_VN (GPIO39), VDET_1 (GPIO34), VDET_2 (GPIO35).
3	The pins are grouped into four power domains: VDDA (analog power supply), VDD3P3_RTC (RTC power supply), VDD3P3_CPU (power supply of digital IOs and CPU cores), VDD_SDIO (power supply of SDIO IOs). VDD_SDIO is the output of the internal SDIO-LDO. The voltage of SDIO-LDO can be configured at 1.8 V or be the same as that of VDD3P3_RTC. The strapping pin and eFuse bits determine the default voltage of the SDIO-LDO. Software can change the voltage of the SDIO-LDO by configuring register bits. For details, please see the column "Power Domain" in Table IO_MUX.
4	The functional pins in the VDD3P3_RTC domain are those with analog functions, including the 32 kHz crystal oscillator, ADC, DAC, and the capacitive touch sensor. Please see columns "Analog Function 1~3" in Table IO_MUX.
5	These VDD3P3_RTC pins support the RTC function, and can work during Deep-sleep. For example, an RTC-GPIO can be used for waking up the chip from Deep-sleep.
6	The GPIO pins support up to six digital functions, as shown in columns "Function 1~6" In Table IO_MUX. The function selection registers will be set as "*N*-1", where *N* is the function number. Below are some definitions: • SD_* is for signals of the SDIO slave. • HS1_* is for Port 1 signals of the SDIO host. • HS2_* is for Port 2 signals of the SDIO host. • MT* is for signals of the JTAG. • U0* is for signals of the UART0 module. • U1* is for signals of the UART1 module. • U2* is for signals of the UART2 module. • SPI* is for signals of the SPI01 module. • HSPI* is for signals of the SPI2 module. • VSPI* is for signals of the SPI3 module.

No.	Description
7	Each column about digital "Function" is accompanied by a column about "Type". Please see the following explanations for the meanings of "type" with respect to each "function" they are associated with. For each "Function-N", "type" signifies: • I: input only. If a function other than "Function-N" is assigned, the input signal of "Function-N" is still from this pin. • I1: input only. If a function other than "Function-N" is assigned, the input signal of "Function-N" is always "1". • I0: input only. If a function other than "Function-N" is assigned, the input signal of "Function-N" is always "0". • O: output only. • T: high-impedance. • I/O/T: combinations of input, output, and high-impedance according to the function signal. • I1/O/T: combinations of input, output, and high-impedance, according to the function signal. If a function is not selected, the input signal of the function is "1". For example, pin 30 can function as HS1_CMD or SD_CMD, where HS1_CMD is of an "I1/O/T" type. If pin 30 is selected as HS1_CMD, this pin's input and output are controlled by the SDIO host. If pin 30 is not selected as HS1_CMD, the input signal of the SDIO host is always "1".
8	Each digital output pin is associated with its configurable drive strength. Column "Drive Strength" in Table IO_MUX lists the default values. The drive strength of the digital output pins can be configured into one of the following four options: • 0: ~5 mA • 1: ~10 mA • 2: ~20 mA • 3: ~40 mA The default value is 2. The drive strength of the internal pull-up (wpu) and pull-down (wpd) is ~75 μA.
9	Column "At Reset" in Table IO_MUX lists the status of each pin during reset, including input-enable (ie=1), internal pull-up (wpu) and internal pull-down (wpd). During reset, all pins are output-disabled.
10	Column "After Reset" in Table IO_MUX lists the status of each pin immediately after reset, including input-enable (ie=1), internal pull-up (wpu) and internal pull-down (wpd). After reset, each pin is set to "Function 1". The output-enable is controlled by digital Function 1.
11	Table Ethernet_MAC is about the signal mapping inside Ethernet MAC. The Ethernet MAC supports MII and RMII interfaces, and supports both the internal PLL clock and the external clock source. For the MII interface, the Ethernet MAC is with/without the TX_ERR signal. MDC, MDIO, CRS and COL are slow signals, and can be mapped onto any GPIO pin through the GPIO-Matrix.
12	Table GPIO Matrix is for the GPIO-Matrix. The signals of the on-chip functional modules can be mapped onto any GPIO pin. Some signals can be mapped onto a pin by both IO-MUX and GPIO-Matrix, as shown in the column tagged as "Same input signal from IO_MUX core" in Table GPIO Matrix.

No.	Description
13	*In Table GPIO_Matrix the column "Default Value if unassigned" records the default value of the an input signal if no GPIO is assigned to it. The actual value is determined by register GPIO_FUNC*m*_IN_INV_SEL and GPIO_FUNC*m*_IN_SEL. (The value of *m* ranges from 1 to 255.)

A.2. GPIO_Matrix

Table 25: GPIO_Matrix

Signal No.	Input signals	Default value if unassigned*	Same input signal from IO_MUX core	Output signals	Output enable of output signals
0	SPICLK_in	0	yes	SPICLK_out	SPICLK_oe
1	SPIQ_in	0	yes	SPIQ_out	SPIQ_oe
2	SPID_in	0	yes	SPID_out	SPID_oe
3	SPIHD_in	0	yes	SPIHD_out	SPIHD_oe
4	SPIWP_in	0	yes	SPIWP_out	SPIWP_oe
5	SPICS0_in	0	yes	SPICS0_out	SPICS0_oe
6	SPICS1_in	0	no	SPICS1_out	SPICS1_oe
7	SPICS2_in	0	no	SPICS2_out	SPICS2_oe
8	HSPICLK_in	0	yes	HSPICLK_out	HSPICLK_oe
9	HSPIQ_in	0	yes	HSPIQ_out	HSPIQ_oe
10	HSPID_in	0	yes	HSPID_out	HSPID_oe
11	HSPICS0_in	0	yes	HSPICS0_out	HSPICS0_oe
12	HSPIHD_in	0	yes	HSPIHD_out	HSPIHD_oe
13	HSPIWP_in	0	yes	HSPIWP_out	HSPIWP_oe
14	U0RXD_in	0	yes	U0TXD_out	1'd1
15	U0CTS_in	0	yes	U0RTS_out	1'd1
16	U0DSR_in	0	no	U0DTR_out	1'd1
17	U1RXD_in	0	yes	U1TXD_out	1'd1
18	U1CTS_in	0	yes	U1RTS_out	1'd1
23	I2S0O_BCK_in	0	no	I2S0O_BCK_out	1'd1
24	I2S1O_BCK_in	0	no	I2S1O_BCK_out	1'd1
25	I2S0O_WS_in	0	no	I2S0O_WS_out	1'd1
26	I2S1O_WS_in	0	no	I2S1O_WS_out	1'd1
27	I2S0I_BCK_in	0	no	I2S0I_BCK_out	1'd1
28	I2S0I_WS_in	0	no	I2S0I_WS_out	1'd1
29	I2CEXT0_SCL_in	1	no	I2CEXT0_SCL_out	1'd1
30	I2CEXT0_SDA_in	1	no	I2CEXT0_SDA_out	1'd1
31	pwm0_sync0_in	0	no	sdio_tohost_int_out	1'd1
32	pwm0_sync1_in	0	no	pwm0_out0a	1'd1
33	pwm0_sync2_in	0	no	pwm0_out0b	1'd1
34	pwm0_f0_in	0	no	pwm0_out1a	1'd1

Signal No.	Input signals	Default value if unassigned	Same input signal from IO_MUX core	Output signals	Output enable of output signals
35	pwm0_f1_in	0	no	pwm0_out1b	1'd1
36	pwm0_f2_in	0	no	pwm0_out2a	1'd1
37	-	0	no	pwm0_out2b	1'd1
39	pcnt_sig_ch0_in0	0	no	-	1'd1
40	pcnt_sig_ch1_in0	0	no	-	1'd1
41	pcnt_ctrl_ch0_in0	0	no	-	1'd1
42	pcnt_ctrl_ch1_in0	0	no	-	1'd1
43	pcnt_sig_ch0_in1	0	no	-	1'd1
44	pcnt_sig_ch1_in1	0	no	-	1'd1
45	pcnt_ctrl_ch0_in1	0	no	-	1'd1
46	pcnt_ctrl_ch1_in1	0	no	-	1'd1
47	pcnt_sig_ch0_in2	0	no	-	1'd1
48	pcnt_sig_ch1_in2	0	no	-	1'd1
49	pcnt_ctrl_ch0_in2	0	no	-	1'd1
50	pcnt_ctrl_ch1_in2	0	no	-	1'd1
51	pcnt_sig_ch0_in3	0	no	-	1'd1
52	pcnt_sig_ch1_in3	0	no	-	1'd1
53	pcnt_ctrl_ch0_in3	0	no	-	1'd1
54	pcnt_ctrl_ch1_in3	0	no	-	1'd1
55	pcnt_sig_ch0_in4	0	no	-	1'd1
56	pcnt_sig_ch1_in4	0	no	-	1'd1
57	pcnt_ctrl_ch0_in4	0	no	-	1'd1
58	pcnt_ctrl_ch1_in4	0	no	-	1'd1
61	HSPICS1_in	0	no	HSPICS1_out	HSPICS1_oe
62	HSPICS2_in	0	no	HSPICS2_out	HSPICS2_oe
63	VSPICLK_in	0	yes	VSPICLK_out_mux	VSPICLK_oe
64	VSPIQ_in	0	yes	VSPIQ_out	VSPIQ_oe
65	VSPID_in	0	yes	VSPID_out	VSPID_oe
66	VSPIHD_in	0	yes	VSPIHD_out	VSPIHD_oe
67	VSPIWP_in	0	yes	VSPIWP_out	VSPIWP_oe
68	VSPICS0_in	0	yes	VSPICS0_out	VSPICS0_oe
69	VSPICS1_in	0	no	VSPICS1_out	VSPICS1_oe
70	VSPICS2_in	0	no	VSPICS2_out	VSPICS2_oe
71	pcnt_sig_ch0_in5	0	no	ledc_hs_sig_out0	1'd1
72	pcnt_sig_ch1_in5	0	no	ledc_hs_sig_out1	1'd1
73	pcnt_ctrl_ch0_in5	0	no	ledc_hs_sig_out2	1'd1
74	pcnt_ctrl_ch1_in5	0	no	ledc_hs_sig_out3	1'd1
75	pcnt_sig_ch0_in6	0	no	ledc_hs_sig_out4	1'd1
76	pcnt_sig_ch1_in6	0	no	ledc_hs_sig_out5	1'd1
77	pcnt_ctrl_ch0_in6	0	no	ledc_hs_sig_out6	1'd1

Signal No.	Input signals	Default value if unassigned	Same input signal from IO_MUX core	Output signals	Output enable of output signals
78	pcnt_ctrl_ch1_in6	0	no	ledc_hs_sig_out7	1'd1
79	pcnt_sig_ch0_in7	0	no	ledc_ls_sig_out0	1'd1
80	pcnt_sig_ch1_in7	0	no	ledc_ls_sig_out1	1'd1
81	pcnt_ctrl_ch0_in7	0	no	ledc_ls_sig_out2	1'd1
82	pcnt_ctrl_ch1_in7	0	no	ledc_ls_sig_out3	1'd1
83	rmt_sig_in0	0	no	ledc_ls_sig_out4	1'd1
84	rmt_sig_in1	0	no	ledc_ls_sig_out5	1'd1
85	rmt_sig_in2	0	no	ledc_ls_sig_out6	1'd1
86	rmt_sig_in3	0	no	ledc_ls_sig_out7	1'd1
87	rmt_sig_in4	0	no	rmt_sig_out0	1'd1
88	rmt_sig_in5	0	no	rmt_sig_out1	1'd1
89	rmt_sig_in6	0	no	rmt_sig_out2	1'd1
90	rmt_sig_in7	0	no	rmt_sig_out3	1'd1
91	-	-	-	rmt_sig_out4	1'd1
92	-	-	-	rmt_sig_out6	1'd1
94	-	-	-	rmt_sig_out7	1'd1
95	I2CEXT1_SCL_in	1	no	I2CEXT1_SCL_out	1'd1
96	I2CEXT1_SDA_in	1	no	I2CEXT1_SDA_out	1'd1
97	host_card_detect_n_1	0	no	host_ccmd_od_pullup_en_n	1'd1
98	host_card_detect_n_2	0	no	host_rst_n_1	1'd1
99	host_card_write_prt_1	0	no	host_rst_n_2	1'd1
100	host_card_write_prt_2	0	no	gpio_sd0_out	1'd1
101	host_card_int_n_1	0	no	gpio_sd1_out	1'd1
102	host_card_int_n_2	0	no	gpio_sd2_out	1'd1
103	pwm1_sync0_in	0	no	gpio_sd3_out	1'd1
104	pwm1_sync1_in	0	no	gpio_sd4_out	1'd1
105	pwm1_sync2_in	0	no	gpio_sd5_out	1'd1
106	pwm1_f0_in	0	no	gpio_sd6_out	1'd1
107	pwm1_f1_in	0	no	gpio_sd7_out	1'd1
108	pwm1_f2_in	0	no	pwm1_out0a	1'd1
109	pwm0_cap0_in	0	no	pwm1_out0b	1'd1
110	pwm0_cap1_in	0	no	pwm1_out1a	1'd1
111	pwm0_cap2_in	0	no	pwm1_out1b	1'd1
112	pwm1_cap0_in	0	no	pwm1_out2a	1'd1
113	pwm1_cap1_in	0	no	pwm1_out2b	1'd1
114	pwm1_cap2_in	0	no	pwm2_out1h	1'd1
115	pwm2_flta	1	no	pwm2_out1l	1'd1
116	pwm2_fltb	1	no	pwm2_out2h	1'd1
117	pwm2_cap1_in	0	no	pwm2_out2l	1'd1
118	pwm2_cap2_in	0	no	pwm2_out3h	1'd1

Signal No.	Input signals	Default value if unassigned	Same input signal from IO_MUX core	Output signals	Output enable of output signals
119	pwm2_cap3_in	0	no	pwm2_out3l	1'd1
120	pwm3_flta	1	no	pwm2_out4h	1'd1
121	pwm3_fltb	1	no	pwm2_out4l	1'd1
122	pwm3_cap1_in	0	no	-	1'd1
123	pwm3_cap2_in	0	no	-	1'd1
124	pwm3_cap3_in	0	no	-	1'd1
140	I2S0I_DATA_in0	0	no	I2S0O_DATA_out0	1'd1
141	I2S0I_DATA_in1	0	no	I2S0O_DATA_out1	1'd1
142	I2S0I_DATA_in2	0	no	I2S0O_DATA_out2	1'd1
143	I2S0I_DATA_in3	0	no	I2S0O_DATA_out3	1'd1
144	I2S0I_DATA_in4	0	no	I2S0O_DATA_out4	1'd1
145	I2S0I_DATA_in5	0	no	I2S0O_DATA_out5	1'd1
146	I2S0I_DATA_in6	0	no	I2S0O_DATA_out6	1'd1
147	I2S0I_DATA_in7	0	no	I2S0O_DATA_out7	1'd1
148	I2S0I_DATA_in8	0	no	I2S0O_DATA_out8	1'd1
149	I2S0I_DATA_in9	0	no	I2S0O_DATA_out9	1'd1
150	I2S0I_DATA_in10	0	no	I2S0O_DATA_out10	1'd1
151	I2S0I_DATA_in11	0	no	I2S0O_DATA_out11	1'd1
152	I2S0I_DATA_in12	0	no	I2S0O_DATA_out12	1'd1
153	I2S0I_DATA_in13	0	no	I2S0O_DATA_out13	1'd1
154	I2S0I_DATA_in14	0	no	I2S0O_DATA_out14	1'd1
155	I2S0I_DATA_in15	0	no	I2S0O_DATA_out15	1'd1
156	-	-	-	I2S0O_DATA_out16	1'd1
157	-	-	-	I2S0O_DATA_out17	1'd1
158	-	-	-	I2S0O_DATA_out18	1'd1
159	-	-	-	I2S0O_DATA_out19	1'd1
160	-	-	-	I2S0O_DATA_out20	1'd1
161	-	-	-	I2S0O_DATA_out21	1'd1
162	-	-	-	I2S0O_DATA_out22	1'd1
163	-	-	-	I2S0O_DATA_out23	1'd1
164	I2S1I_BCK_in	0	no	I2S1I_BCK_out	1'd1
165	I2S1I_WS_in	0	no	I2S1I_WS_out	1'd1
166	I2S1I_DATA_in0	0	no	I2S1O_DATA_out0	1'd1
167	I2S1I_DATA_in1	0	no	I2S1O_DATA_out1	1'd1
168	I2S1I_DATA_in2	0	no	I2S1O_DATA_out2	1'd1
169	I2S1I_DATA_in3	0	no	I2S1O_DATA_out3	1'd1
170	I2S1I_DATA_in4	0	no	I2S1O_DATA_out4	1'd1
171	I2S1I_DATA_in5	0	no	I2S1O_DATA_out5	1'd1
172	I2S1I_DATA_in6	0	no	I2S1O_DATA_out6	1'd1
173	I2S1I_DATA_in7	0	no	I2S1O_DATA_out7	1'd1

Signal No.	Input signals	Default value if unassigned	Same input signal from IO_MUX core	Output signals	Output enable of output signals
174	I2S1I_DATA_in8	0	no	I2S1O_DATA_out8	1'd1
175	I2S1I_DATA_in9	0	no	I2S1O_DATA_out9	1'd1
176	I2S1I_DATA_in10	0	no	I2S1O_DATA_out10	1'd1
177	I2S1I_DATA_in11	0	no	I2S1O_DATA_out11	1'd1
178	I2S1I_DATA_in12	0	no	I2S1O_DATA_out12	1'd1
179	I2S1I_DATA_in13	0	no	I2S1O_DATA_out13	1'd1
180	I2S1I_DATA_in14	0	no	I2S1O_DATA_out14	1'd1
181	I2S1I_DATA_in15	0	no	I2S1O_DATA_out15	1'd1
182	-	-	-	I2S1O_DATA_out16	1'd1
183	-	-	-	I2S1O_DATA_out17	1'd1
184	-	-	-	I2S1O_DATA_out18	1'd1
185	-	-	-	I2S1O_DATA_out19	1'd1
186	-	-	-	I2S1O_DATA_out20	1'd1
187	-	-	-	I2S1O_DATA_out21	1'd1
188	-	-	-	I2S1O_DATA_out22	1'd1
189	-	-	-	I2S1O_DATA_out23	1'd1
190	I2S0I_H_SYNC	0	no	pwm3_out1h	1'd1
191	I2S0I_V_SYNC	0	no	pwm3_out1l	1'd1
192	I2S0I_H_ENABLE	0	no	pwm3_out2h	1'd1
193	I2S1I_H_SYNC	0	no	pwm3_out2l	1'd1
194	I2S1I_V_SYNC	0	no	pwm3_out3h	1'd1
195	I2S1I_H_ENABLE	0	no	pwm3_out3l	1'd1
196	-	-	-	pwm3_out4h	1'd1
197	-	-	-	pwm3_out4l	1'd1
198	U2RXD_in	0	yes	U2TXD_out	1'd1
199	U2CTS_in	0	yes	U2RTS_out	1'd1
200	emac_mdc_i	0	no	emac_mdc_o	emac_mdc_oe
201	emac_mdi_i	0	no	emac_mdo_o	emac_mdo_o_e
202	emac_crs_i	0	no	emac_crs_o	emac_crs_oe
203	emac_col_i	0	no	emac_col_o	emac_col_oe
204	pcmfsync_in	0	no	bt_audio0_irq	1'd1
205	pcmclk_in	0	no	bt_audio1_irq	1'd1
206	pcmdin	0	no	bt_audio2_irq	1'd1
207	-	-	-	ble_audio0_irq	1'd1
208	-	-	-	ble_audio1_irq	1'd1
209	-	-	-	ble_audio2_irq	1'd1
210	-	-	-	pcmfsync_out	pcmfsync_en
211	-	-	-	pcmclk_out	pcmclk_en
212	-	-	-	pcmdout	pcmdout_en
213	-	-	-	ble_audio_sync0_p	1'd1

Signal No.	Input signals	Default value if unassigned	Same input signal from IO_MUX core	Output signals	Output enable of output signals
214	-	-	-	ble_audio_sync1_p	1'd1
215	-	-	-	ble_audio_sync2_p	1'd1
224	-	-	-	sig_in_func224	1'd1
225	-	-	-	sig_in_func225	1'd1
226	-	-	-	sig_in_func226	1'd1
227	-	-	-	sig_in_func227	1'd1
228	-	-	-	sig_in_func228	1'd1

A.3. Ethernet_MAC

Table 26: Ethernet_MAC

PIN Name	Function6	MII (int_osc)	MII (ext_osc)	RMII (int_osc)	RMII (ext_osc)
GPIO0	EMAC_TX_CLK	TX_CLK (I)	TX_CLK (I)	CLK_OUT(O)	EXT_OSC_CLK(I)
GPIO5	EMAC_RX_CLK	RX_CLK (I)	RX_CLK (I)	-	-
GPIO21	EMAC_TX_EN	TX_EN(O)	TX_EN(O)	TX_EN(O)	TX_EN(O)
GPIO19	EMAC_TXD0	TXD[0](O)	TXD[0](O)	TXD[0](O)	TXD[0](O)
GPIO22	EMAC_TXD1	TXD[1](O)	TXD[1](O)	TXD[1](O)	TXD[1](O)
MTMS	EMAC_TXD2	TXD[2](O)	TXD[2](O)	-	-
MTDI	EMAC_TXD3	TXD[3](O)	TXD[3](O)	-	-
MTCK	EMAC_RX_ER	RX_ER(I)	RX_ER(I)	-	-
GPIO27	EMAC_RX_DV	RX_DV(I)	RX_DV(I)	CRS_DV(I)	CRS_DV(I)
GPIO25	EMAC_RXD0	RXD[0](I)	RXD[0](I)	RXD[0](I)	RXD[0](I)
GPIO26	EMAC_RXD1	RXD[1](I)	RXD[1](I)	RXD[1](I)	RXD[1](I)
U0TXD	EMAC_RXD2	RXD[2](I)	RXD[2](I)	-	-
MTDO	EMAC_RXD3	RXD[3](I)	RXD[3](I)	-	-
GPIO16	EMAC_CLK_OUT	CLK_OUT(O)	-	CLK_OUT(O)	-
GPIO17	EMAC_CLK_OUT_180	CLK_OUT_180(O)	-	CLK_OUT_180(O)	-
GPIO4	EMAC_TX_ER	TX_ERR(O)*	TX_ERR(O)*	-	-
In GPIO Matrix*	-	MDC(O)	MDC(O)	MDC(O)	MDC(O)
In GPIO Matrix*	-	MDIO(IO)	MDIO(IO)	MDIO(IO)	MDIO(IO)
In GPIO Matrix*	-	CRS(I)	CRS(I)	-	-
In GPIO Matrix*	-	COL(I)	COL(I)	-	-
*Notes: 1. The GPIO Matrix can be any GPIO. 2. The TX_ERR (O) is optional.					

A.4. IO_MUX

For the list of IO_MUX pins, please see the next page.

IO_MUX

Pin No.	Power Supply Pin	Analog Pin	Digital Pin	Power Domain	Analog Function1	Analog Function2	Analog Function3	RTC Function1	RTC Function2	Function1	Type	Function2	Type	Function3	Type	Function4	Type	Function5	Type	Function6	Type	Drive Strength (2'd2: 20 mA)	At Reset	After Reset
1	VDDA			VDDA supply in																				
2		LNA_IN		VDD3P3																				
3	VDD3P3			VDD3P3 supply in																				
4	VDD3P3			VDD3P3 supply in																				
5		SENSOR_VP		VDD3P3_RTC	ADC_H	ADC1_CH0		RTC_GPIO0		GPIO36	I			GPIO36	I								oe=0, ie=0	oe=0, ie=0
6		SENSOR_CAPP		VDD3P3_RTC	ADC_H	ADC1_CH1		RTC_GPIO1		GPIO37	I			GPIO37	I								oe=0, ie=0	oe=0, ie=0
7		SENSOR_CAPN		VDD3P3_RTC	ADC_H	ADC1_CH2		RTC_GPIO2		GPIO38	I			GPIO38	I								oe=0, ie=0	oe=0, ie=0
8		SENSOR_VN		VDD3P3_RTC	ADC_H	ADC1_CH3		RTC_GPIO3		GPIO39	I			GPIO39	I								oe=0, ie=0	oe=0, ie=0
9		CHIP_PU		VDD3P3_RTC																				
10		VDET_1		VDD3P3_RTC		ADC1_CH6		RTC_GPIO4		GPIO34	I			GPIO34	I								oe=0, ie=0	oe=0, ie=0
11		VDET_2		VDD3P3_RTC		ADC1_CH7		RTC_GPIO5		GPIO35	I			GPIO35	I								oe=0, ie=0	oe=0, ie=0
12		32K_XP		VDD3P3_RTC	XTAL_32K_P	ADC1_CH4	TOUCH9	RTC_GPIO9		GPIO32	I/O/T			GPIO32	I/O/T							2'd2	oe=0, ie=0	oe=0, ie=0
13		32K_XN		VDD3P3_RTC	XTAL_32K_N	ADC1_CH5	TOUCH8	RTC_GPIO8		GPIO33	I/O/T			GPIO33	I/O/T							2'd2	oe=0, ie=0	oe=0, ie=0
14			GPIO25	VDD3P3_RTC	DAC_1	ADC2_CH8		RTC_GPIO6		GPIO25	I/O/T			GPIO25	I/O/T					EMAC_RXD0	I	2'd2	oe=0, ie=0	oe=0, ie=0
15			GPIO26	VDD3P3_RTC	DAC_2	ADC2_CH9		RTC_GPIO7		GPIO26	I/O/T			GPIO26	I/O/T					EMAC_RXD1	I	2'd2	oe=0, ie=0	oe=0, ie=0
16			GPIO27	VDD3P3_RTC		ADC2_CH7	TOUCH7	RTC_GPIO17		GPIO27	I/O/T			GPIO27	I/O/T					EMAC_RX_DV	I	2'd2	oe=0, ie=0	oe=0, ie=1
17			MTMS	VDD3P3_RTC		ADC2_CH6	TOUCH6	RTC_GPIO16		MTMS	I0	HSPICLK	I/O/T	GPIO14	I/O/T	HS2_CLK	O	SD_CLK	I0	EMAC_TXD2	O	2'd2	oe=0, ie=0	oe=0, ie=1
18			MTDI	VDD3P3_RTC		ADC2_CH5	TOUCH5	RTC_GPIO15		MTDI	I1	HSPIQ	I/O/T	GPIO12	I/O/T	HS2_DATA2	I1/O/T	SD_DATA2	I1/O/T	EMAC_TXD3	O	2'd2	oe=0, ie=1, wpd	oe=0, ie=1, wpd
19	VDD3P3_RTC			VDD3P3_RTC supply in																				
20			MTCK	VDD3P3_RTC		ADC2_CH4	TOUCH4	RTC_GPIO14		MTCK	I1	HSPID	I/O/T	GPIO13	I/O/T	HS2_DATA3	I1/O/T	SD_DATA3	I1/O/T	EMAC_RX_ER	I	2'd2	oe=0, ie=0	oe=0, ie=1
21			MTDO	VDD3P3_RTC		ADC2_CH3	TOUCH3	RTC_GPIO13	I2C_SDA	MTDO	O/T	HSPICS0	I/O/T	GPIO15	I/O/T	HS2_CMD	I1/O/T	SD_CMD	I1/O/T	EMAC_RXD3	I	2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
22			GPIO2	VDD3P3_RTC		ADC2_CH2	TOUCH2	RTC_GPIO12	I2C_SCL	GPIO2	I/O/T	HSPIWP	I/O/T	GPIO2	I/O/T	HS2_DATA0	I1/O/T	SD_DATA0	I1/O/T			2'd2	oe=0, ie=1, wpd	oe=0, ie=1, wpd
23			GPIO0	VDD3P3_RTC		ADC2_CH1	TOUCH1	RTC_GPIO11	I2C_SDA	GPIO0	I/O/T	CLK_OUT1	O	GPIO0	I/O/T					EMAC_TX_CLK	I	2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
24			GPIO4	VDD3P3_RTC		ADC2_CH0	TOUCH0	RTC_GPIO10	I2C_SCL	GPIO4	I/O/T	HSPIHD	I/O/T	GPIO4	I/O/T	HS2_DATA1	I1/O/T	SD_DATA1	I1/O/T	EMAC_TX_ER	O	2'd2	oe=0, ie=1, wpd	oe=0, ie=1, wpd
25			GPIO16	VDD_SDIO						GPIO16	I/O/T			GPIO16	I/O/T	HS1_DATA4	I1/O/T	U2RXD	I1	EMAC_CLK_OUT	O	2'd2	oe=0, ie=0	oe=0, ie=1
26	VDD_SDIO			VDD_SDIO supply out/in																				
27			GPIO17	VDD_SDIO						GPIO17	I/O/T			GPIO17	I/O/T	HS1_DATA5	I1/O/T	U2TXD	O	EMAC_CLK_OUT_180	O	2'd2	oe=0, ie=0	oe=0, ie=1
28			SD_DATA_2	VDD_SDIO						SD_DATA2	I1/O/T	SPIHD	I/O/T	GPIO9	I/O/T	HS1_DATA2	I1/O/T	U1RXD	I1			2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
29			SD_DATA_3	VDD_SDIO						SD_DATA3	I0/O/T	SPIWP	I/O/T	GPIO10	I/O/T	HS1_DATA3	I1/O/T	U1TXD	O			2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
30			SD_CMD	VDD_SDIO						SD_CMD	I1/O/T	SPICS0	I/O/T	GPIO11	I/O/T	HS1_CMD	I1/O/T	U1RTS	O			2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
31			SD_CLK	VDD_SDIO						SD_CLK	I0	SPICLK	I/O/T	GPIO6	I/O/T	HS1_CLK	O	U1CTS	I1			2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
32			SD_DATA_0	VDD_SDIO						SD_DATA0	I1/O/T	SPIQ	I/O/T	GPIO7	I/O/T	HS1_DATA0	I1/O/T	U2RTS	O			2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
33			SD_DATA_1	VDD_SDIO						SD_DATA1	I1/O/T	SPID	I/O/T	GPIO8	I/O/T	HS1_DATA1	I1/O/T	U2CTS	I1			2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
34			GPIO5	VDD3P3_CPU						GPIO5	I/O/T	VSPICS0	I/O/T	GPIO5	I/O/T	HS1_DATA6	I1/O/T			EMAC_RX_CLK	I	2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
35			GPIO18	VDD3P3_CPU						GPIO18	I/O/T	VSPICLK	I/O/T	GPIO18	I/O/T	HS1_DATA7	I1/O/T					2'd2	oe=0, ie=0	oe=0, ie=1
36			GPIO23	VDD3P3_CPU						GPIO23	I/O/T	VSPID	I/O/T	GPIO23	I/O/T	HS1_STROBE	I0					2'd2	oe=0, ie=0	oe=0, ie=1
37	VDD3P3_CPU			VDD3P3_CPU supply in																				
38			GPIO19	VDD3P3_CPU						GPIO19	I/O/T	VSPIQ	I/O/T	GPIO19	I/O/T	U0CTS	I1			EMAC_TXD0	O	2'd2	oe=0, ie=0	oe=0, ie=1
39			GPIO22	VDD3P3_CPU						GPIO22	I/O/T	VSPIWP	I/O/T	GPIO22	I/O/T	U0RTS	O			EMAC_TXD1	O	2'd2	oe=0, ie=0	oe=0, ie=1
40			U0RXD	VDD3P3_CPU						U0RXD	I1	CLK_OUT2	O	GPIO3	I/O/T							2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
41			U0TXD	VDD3P3_CPU						U0TXD	O	CLK_OUT3	O	GPIO1	I/O/T					EMAC_RXD2	I	2'd2	oe=0, ie=1, wpu	oe=0, ie=1, wpu
42			GPIO21	VDD3P3_CPU						GPIO21	I/O/T	VSPIHD	I/O/T	GPIO21	I/O/T					EMAC_TX_EN	O	2'd2	oe=0, ie=0	oe=0, ie=1
43	VDDA			VDDA supply in																				
44		XTAL_N		VDDA																				
45		XTAL_P		VDDA																				
46	VDDA			VDDA supply in																				
47		CAP2		VDDA																				
48		CAP1		VDDA																				
Total Number	8	14	26																					

Notes:

- wpu: weak pull-up;
- wpd: weak pull-down;
- ie: input enable;
- oe: output enable;
- Please see *Table: Notes on ESP32 Pin Lists* for more information.（请参考表：管脚清单说明。）

Revision History

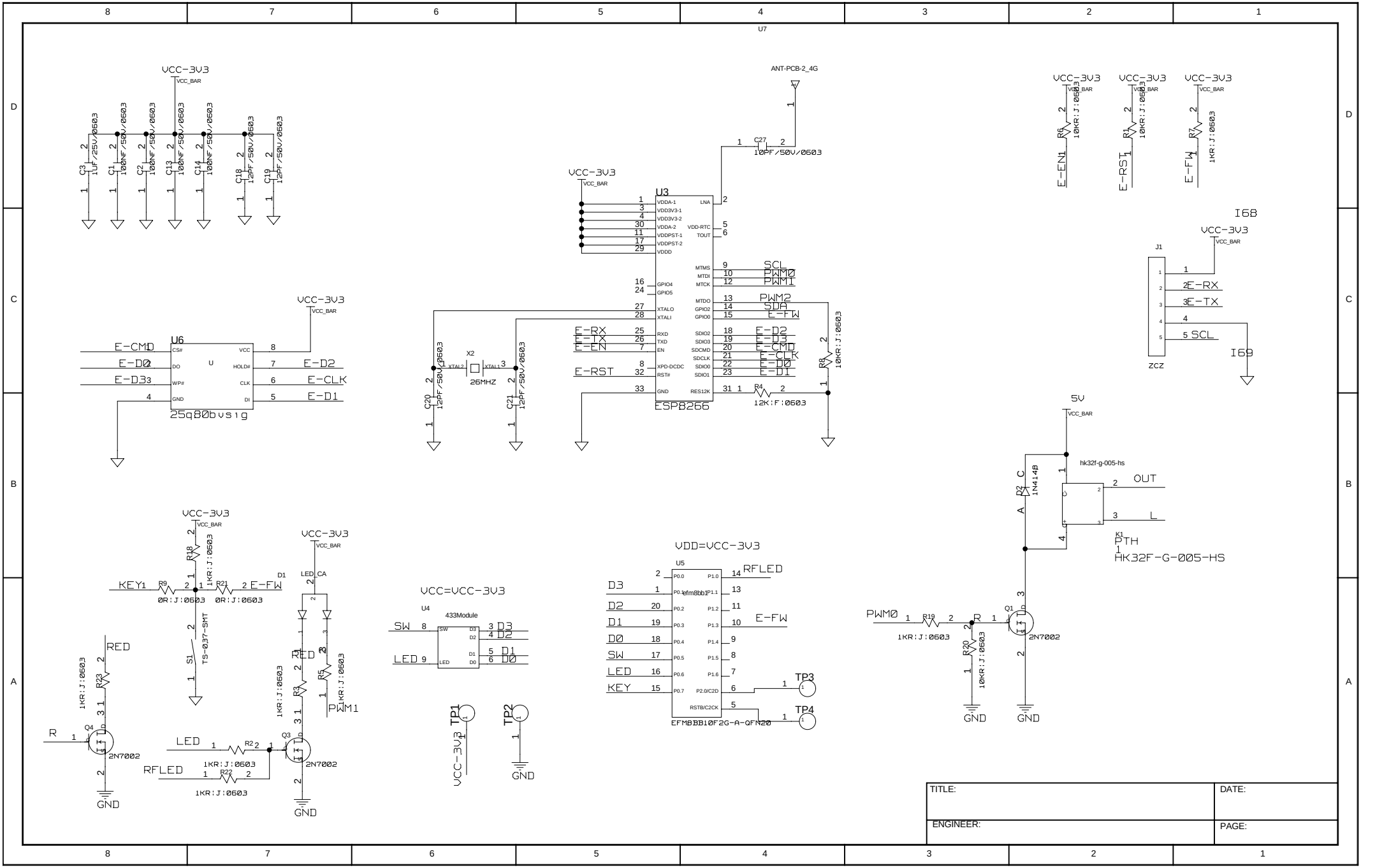
Date	Version	Release notes
2020-04-27	V3.4	Added one chip variant: ESP32-U4WDH Updated some figures in Table 6, 16, 17, 19, 21, 22 Added a note under Table 18
2020.01	V3.3	Added two chip variants: ESP32-D0WD-V3 and ESP32-D0WDQ6-V3. Added a note under Table 7.
2019.10	V3.2	Updated Figure 5: *ESP32 Power-up and Reset Timing*.
2019.07	V3.1	Added pin-pin mapping between ESP32-D2WD and the embedded flash under Table 1 *Pin Description*; Updated Figure 10 *ESP32 Part Number*.
2019.04	V3.0	Added information about the setup and hold times for the strapping pins in Section 2.4: Strapping Pins.
2019.02	V2.9	Applied new formatting to Table 1: Pin Description; Fixed typos with respect to the ADC1 channel mappings in Table 10: Peripheral Pin Configurations.
2019.01	V2.8	Changed the RF power control range in Table 18, Table 20 and Table 22 from –12 ~ +12 to –12 ~ +9 dBm; Small text changes.
2018.11	V2.7	Updated Section 1.5; Updated pin statuses at reset and after reset in Table IO_MUX.
2018.10	V2.6	Updated QFN package drawings in Chapter 6: Package Information.
2018.08	V2.5	• Added ”Cumulative IO output current” entry to Table 11: Absolute Maximum Ratings; • Added more parameters to Table 13: DC Characteristics; • Changed the power domain names in Table IO_MUX to be consistent with the pin names.
2018.07	V2.4	• Deleted information on Packet Traffic Arbitration (PTA); • Added Figure 5: ESP32 Power-up and Reset Timing in Section 2.3: Power Scheme; • Added the power consumption of dual-core SoCs in Table 6: Power Consumption by Power Modes; • Updated section 4.1.2: Analog-to-Digital Converter (ADC).
2018.06	V2.3	Added the power consumption at CPU frequency of 160 MHz in Table 6: Power Consumption by Power Modes.

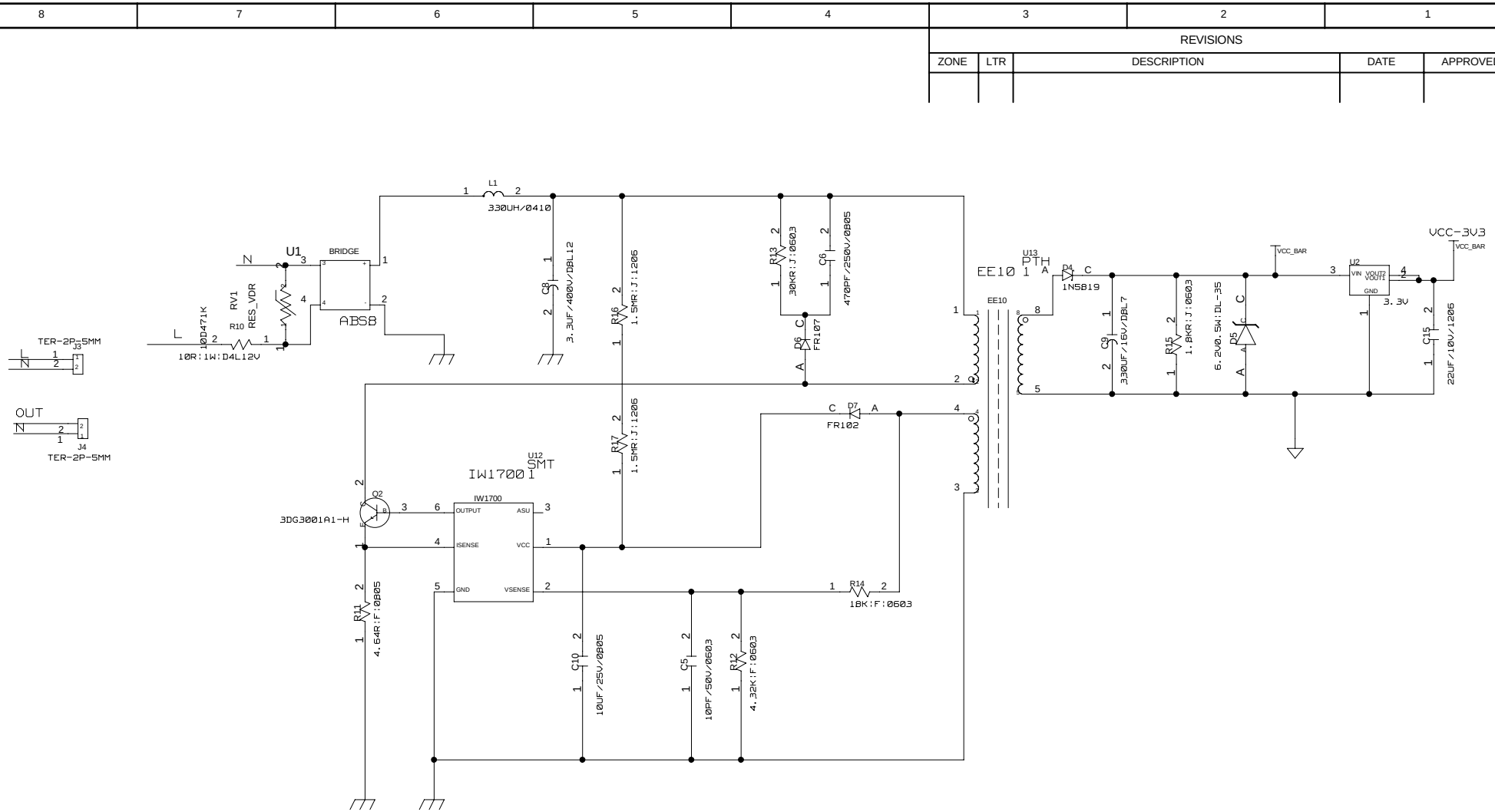
Date	Version	Release notes
2018.05	V2.2	• Changed the voltage range of VDD3P3_RTC from 1.8-3.6V to 2.3-3.6V in Table 1: Pin Description; • Updated Section 2.3: Power Scheme; • Updated Section 3.1.3: External Flash and SRAM; • Updated Table 6: Power Consumption by Power Modes; • Deleted content about temperature sensor; Changes to electrical characteristics: • Updated Table 11: Absolute Maximum Ratings; • Added Table 12: Recommended Operating Conditions; • Added Table 13: DC Characteristics; • Added Table 14: Reliability Qualifications; • Updated the values of "Gain control step" and "Adjacent channel transmit power" in Table 18: Transmitter Characteristics - Basic Data Rate; • Updated the values of "Gain control step", "π/4 DQPSK modulation accuracy", "8 DPSK modulation accuracy" and "In-band spurious emissions" in Table 20: Transmitter Characteristics – Enhanced Data Rate; • Updated the values of "Gain control step", "Adjacent channel transmit power" in Table 22: Transmitter Characteristics - BLE.
2018.01	V2.1	• Deleted software-specific features; • Deleted information on LNA pre-amplifier; • Specified the CPU speed and flash speed of ESP32-D2WD; • Added notes to Section 2.3: Power Scheme.
2017.12	V2.0	Added a note on the sequence of pin number in Chapter 6.
2017.10	V1.9	• Updated the description of the pin CHIP_PU in Table 1; • Added a note to Section 2.3: Power Scheme; • Updated the description of the chip's system reset in Section 2.4: Strapping Pins; • Added a description of antenna diversity and selection to Section 3.5.1; • Deleted "Association sleep pattern" in Table 6 and added notes to Active sleep and Modem-sleep.
2017.08	V1.8	• Added Table 4.2 in Section 4; • Corrected a typo in Figure 1.

Date	Version	Release notes
2017.08	V1.7	• Changed the transmitting power to +12 dBm; the sensitivity of NZIF receiver to -97 dBm in Section 1.3; • Added a note to Table 1 Pin Description; • Added 160 MHz clock frequency in section 3.1.1; • Changed the transmitting power from 21 dBm to 20.5 dBm in Section 3.5.1; • Changed the dynamic control range of class-1, class-2 and class-3 transmit output powers to ”up to 24 dBm”; and changed the dynamic range of NZIF receiver sensitivity to ”over 97 dB” in Section 3.6.1; • Updated Table 6: Power Consumption by Power Modes, and added two notes to it; • Updated sections 4.1.1, 4.1.9; • Updated Table 11: Absolute Maximum Ratings; • Updated Table 15: RF Power Consumption Specifications, and changed the duty cycle on which the transmitters’ measurements are based by 50%. • Updated Table 16: Wi-Fi Radio Characteristics and added a note on “Output impedance” to it; • Updated parameter ”Sensitivity” in Table 17, 19, 21; • Updated parameters ”RF transmit power” and ”RF power control range”, and added parameter ”Gain control step” in Table 18, 20, 22; • Deleted Chapters: ”Touch Sensor” and ”Code Examples”; • Added a link to certification download.
2017.06	V1.6	Corrected two typos: • Changed the number of external components to 20 in Section 1.1.2; • Changed the number of GPIO pins to 34 in Section 4.1.1.
2017.06	V1.5	• Changed the power supply range in Section: 1.4.1 CPU and Memory; • Updated the note in Section 2.3: Power Scheme; • Updated Table 11: Absolute Maximum Ratings; • Changed the drive strength values of the digital output pins in Note 8, in Table 24: Notes on ESP32 Pin Lists; • Added the option to subscribe for notifications of documentation changes.
2017.05	V1.4	• Added a note to the frequency of the external crystal oscillator in Section 1.4.2: Clocks and Timers; • Added a note to Section 2.4: Strapping Pins; • Updated Section 3.7: RTC and Low-Power Management; • Changed the maximum driving capability from 12 mA to 80 mA, in Table 11: Absolulte Maximum Ratings; • Changed the input impedance value of 50Ω, in Table 16: Wi-Fi Radio Characteristics, to output impedance value of 30+j10 Ω; • Added a note to No.8 in Table 24: Notes on ESP32 Pin Lists; • Deleted GPIO20 in Table IO_MUX.
2017.04	V1.3	• Added Appendix: ESP32 Pin Lists; • Updated Table: Wi-Fi Radio Characteristics; • Updated Figure: ESP32 Pin Layout (for QFN 5*5).

Date	Version	Release notes
2017.03	V1.2	• Added a note to Table: Pin Description; • Updated the note in Section: Internal Memory.
2017.02	V1.1	• Added Chapter: Part Number and Ordering Information; • Updated Section: MCU and Advanced Features; • Updated Section: Block Diagram; • Updated Chapter: Pin Definitions; • Updated Section: CPU and Memory; • Updated Section: Audio PLL Clock; • Updated Section: Absolute Maximum Ratings; • Updated Chapter: Package Information; • Updated Chapter: Learning Resources.
2016.08	V1.0	First release.

APPENDIX H: SONOFF SCHEMATICS





Campus de Rabanales, Edificio Leonardo Da Vinci
14071 CÓRDOBA (España)
Teléfono 957-218373