

Design and Implementation of the Universal RTOS on DSP

JIANG Hong, YU Qing-song, WANG Xi-lin

Computer Center
East China Normal University
Shanghai, P. R. of China
hjiang@cc.ecnu.edu.cn

Abstract—A real-time operating system GDOS (Generic DSP Operating System) based on DSP is designed and implemented in this paper. GDOS can solve the disunity of current DSP interfaces and improve the portability of the application systems on GDOS. Furthermore, GDOS can improve software engineers' efficiency. GDOS has been successfully applied in a 3G radio network controller. This 3G radio network controller is currently working steadily at home and abroad.

Keywords: *DSP; Real-time Operating Systems; Task; Synchronization; Mutex; Memory Management; Log; Portability*

I. INTRODUCTION

DSP chips have developed rapidly since 1980. With the continuous improvement of price-performance and development methods of DSP chips^[1, 2], DSP has been widely used in various fields such as communications, audio, video, graphics, images, military, automation, home appliances, and automotive electronic systems.

Many manufacturers have developed their own DSP application systems and DSP real-time operating systems (RTOS), e.g., DSP BIOS of Texas Instruments^[3] and SmartDSP OS of Freescale^[4]. RTOS has reduced the complexity of DSP software development, and has played an important role in promoting the popularity and development of DSP technology.

However, DSP RTOS provided by different manufacturers differs in external interfaces. Even the different types of DSP RTOS provided by the same manufacturers have different external interfaces. DSP software developers must be familiar with the different external interfaces of DSP RTOS. System transplantation will be rather time-consuming when DSP applications are required in different DSP platforms.

Moreover, the allocation and release of dynamic memory in current DSP RTOS are carried out directly in the memory. It will produce more memory fragmentation after the system runs for a long time. Memory fragmentation requires extra memory management overhead.

In addition, current DSP RTOS lacks a unified log function to support multiple output forms. It leads to great inconvenience when collecting log information during

debugging and running time. It also increases the difficulty of system maintenance.

A generic DSP real-time operating system (GDOS) is designed and implemented in this paper. Apart from the common functions of operating system^[5], GDOS has the following characteristics:

- Portability: GDOS provides a consistent application interface which is compatible with different underlying platforms;
- Stability: GDOS provides enhanced memory management functions to address memory fragmentation problems in the RTOS;
- Maintainability: GDOS provides powerful log management functions to improve system maintainability;
- Debugging functionality: GDOS provides debugging functions for each functional module to make the system stabilize more quickly.

The remainder of this paper is structured as follows. The design and implementation of GDOS is presented in detail in Section 2. The performance of GDOS is tested and analyzed in Section 3. Section 4 provides a conclusion.

II. DESIGN AND IMPLEMENTATION OF GDOS

Based on the RTOS provided by DSP chip manufacturers, GDOS provides a unified interface for different application systems, shielding application systems from the impact of the transformation of underlying platform. GDOS provides such functional modules as multi-task management, clock management, synchronization and mutex, memory management, log management, and debugging function. Figure 1 shows the structure of GDOS.

A. GDOS Multi-task Management Function

GDOS is a multi-task RTOS. Multi-task management function includes task creating and task scheduling. A task registry table is defined in GDOS. After the power is on, according to the task registry table GDOS creates and schedules the tasks by invoking the underlying interface commands. Application systems are only required to provide

such information as task entry function, task priority, and stack size of the task registry table, but not needed to care about the details of task creation and scheduling.

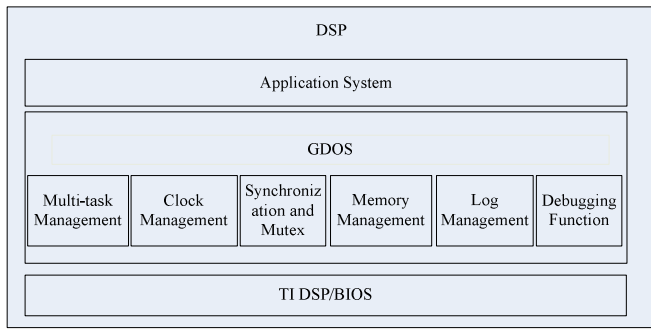


Figure 1. GDOS Structure^[6]

B. GDOS Clock Management Function

GDOS's clock management is based on DSP/BIOS clock interruption function. Timer interruption routine is associated with DSP/BIOS system clock. In the timer interruption (with 1ms cycle) routine, the counter of 1ms is accumulated, which provides 1ms time accuracy.

There exist two kinds of timer in GDOS: one is implemented by system clock interruption with millisecond accuracy. As this timer provides the callback mechanism, we call it 'callback timer'; the other does not support timer callback function. It detects whether the timer is overtime by calling GDOS interface function, so we called it 'extended timer'^[7]. The accuracy of the extended timer depends on the accuracy of the system clock (in milliseconds) and the frequency of the application system to detect whether the timer is overtime. Extended timer is suitable for loop scheduling. It has the advantage of lower system overhead.

C. GDOS Synchronization and Mutex Function

GDOS provides synchronization and mutex function for a common resource that many tasks access simultaneously: when there is no access to the shared resource, one task is permitted to access it; when a task is currently accessing the shared resource, other tasks attempting to access the shared resource will have to wait; when a task finishes accessing the shared resources, another task in the waiting queue is permitted to access it.

GDOS provides two kinds of synchronization and mutex function which can meet the different requirements of application systems: lock variables and semaphores^[8].

GDOS's lock variables are implemented by the DSP/BIOS interface. Through encapsulation on the bottom functions, GDOS provides a unified interface for the lock variables, including creation, deletion, locking and unlocking of lock variables.

GDOS's semaphores are also implemented by the DSP/BIOS interface. Through encapsulation on the bottom function, GDOS provides a unified interface for the

semaphores, including creation, P operation, V operation and counter retrieving of semaphores.

D. GDOS Memory Management Function

In order to achieve rapid and efficient memory allocation and management functions, GDOS manages memory by memory pool.

The core idea of memory pool is to allocate all the memory to predefined small memory blocks before GDOS starts its tasks. Memory blocks are managed through doubly linked lists. When application system requests memory, a suitable memory block from the doubly linked list is assigned to it; whereas the memory block is returned to the doubly linked list when the application system releases memory for re-allocation in future.

GDOS provides a configuration table for memory pool. During the system is powering up, memory pool is initialized according to the configuration table. Memory blocks of the same size are managed by the same doubly linked list. Each memory block assigned to the user has a corresponding memory head to store the debugging and management information of the memory block.

E. GDOS Log Management Function

Log function is an important part of a sophisticated system. Convenient and efficient log functions will facilitate debugging and maintaining of the system and reduce the corresponding manpower cost. GDOS's log management function has the following features:

- **Controllability:** GDOS provides different log levels for different functional modules. The name and ID of the module are registered in GDOS. There exist five log levels: DETAIL, MSG, WARN, ERROR, and FATAL.
- **Diversity of output forms:** If the programs are debugged through the simulator, logs are exported to the integrated environment CCS on PC through JTAG port and displayed on the STD window on CCS. Whereas the programs are not debugged through the simulator, logs are export to the host CPU through HPI port. The host CPU will display the logs in a suitable form according to the actual situation.
- **Lower overhead:** GDOS can minimize memory overhead as well as scheduling overhead of DSP.

F. GDOS Debugging Function

Debugging is an inevitable function in a business system. Debugging function is directly related to the speed of system stability during R&D process, as well as the difficulty and the workload of system maintenance. GDOS provides debugging functions for task management, memory management, and log management.

GDOS's multi-task management debugging function is mainly used to record the status information when abnormal malfunction such as DSP halting occurs. GDOS also records task switching information to satisfy the debugging requirements of multi-task management.

GDOS's memory management debugging function is mainly used for memory statistics and memory write protection. GDOS's memory statistics function allows the users to inquire the allocation status of current system memory (the amount of occupied memory and available memory, etc.) and identify whether there exists memory leak.

GDOS's log management debugging function helps application systems to export log information for future debugging. Log information is exported through HPI port. Log information may be lost in emergency situation. Therefore log information is numbered in GDOS. Users can determine whether the log information is lost according to whether the number is continuous. In addition, system time is also automatically recorded in GDOS's log information to help the users to analyze log information.

III. SYSTEM TEST AND PERFORMANCE ANALYSIS OF GDOS

System test and performance analysis are the key steps to verify whether system design and implementation satisfy system requirements. Test cases are designed in this paper to verify whether the functions of GDOS fulfill the business requirements and user expectations.

GDOS provides system test and performance analysis of log management, multi-task management, clock management, and memory management.

A. Test and Analysis of Log Management Function

GDOS's log management function provides two kinds of output forms: output through JTAG port and output to host CPU through HPI port.

After log information is exported to the host through HPI port, the host exports log information to background through UDP. A tool called UdpWatch is designed to collect log information. UdpWatch is running on PC, receiving log information from the host and supporting sub-module log level configuration.

After log information exports onto UdpWatch, such information as module name, DSP number, DSP log number, DSP time, and log content (ref. Fig. 2) will be displayed.

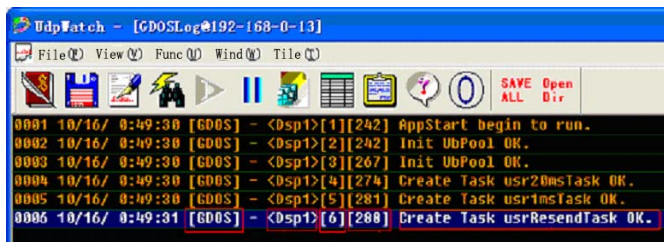


Figure 2. Log output results

It shows that log information can be exported to UdpWatch through GDOS's log management function. With UdpWatch the users can easily view the log information and set the log level.

B. Test and Analysis of Multi-task Management Function

In order to test GDOS's multi-task management function, three processes are registered in GDOS's task registry table: usr20msTask, usr1msTask, and usrResendTask, with priority 3, 6, 7 respectively. After the system runs, GDOS's task query results are shown in Fig. 3.

```

-> ShowGDOSTaskInfo 1

```

DspNo	TaskName	Priority	StackSize	Status
1	usr20msTask	3	8192	OK
1	usr1msTask	6	4096	OK
1	usrResendTask	7	2048	OK

```

value = 64 = 0x40 = 'd'
->

```

Figure 3. Test results of GDOS task query

Fig. 3 shows that the system has three application system tasks. The normal state ("OK") indicates that the three tasks have been created successfully and have already been scheduled normally.

C. Test and Analysis of Clock Management Function

Test cases should be designed to test GDOS's clock management function. Our test case creates a callback-supported timer. The timer overtime callback function will display current system time and related prompt information to determine whether the timer functions or not. Fig. 4 shows the test results. The timer TestTimer1 starts at 610520ms with the length of 1000ms. TestTimer1's timeout occurs at 611520ms, 612520ms, and 613520ms respectively.

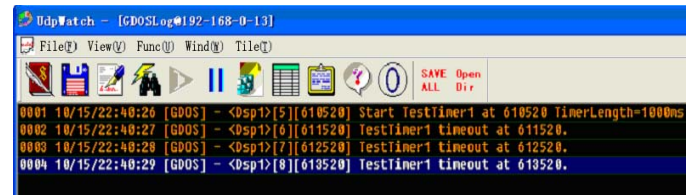


Figure 4. Test results of GDOS clock management

Fig. 4 shows that GDOS's system time query function and timer function run normally. These functions have met the system requirements and achieved the desired results.

D. Test and Analysis of Memory Management Function

GDOS's memory management function provides a memory information query function which queries such information as type of memory pool, memory block size of each memory pool, amount of total memory blocks, amount of allocated memory blocks, and amount of free memory blocks, etc.

```

-> ShowGDOSUbiInfo 1

```

UbiPoolNo	UbiSize	TotalNum	UsedNum	FreeNum
1	512	400	1	399
2	1024	600	1	599
3	2048	600	0	600
4	4096	600	0	600
5	8192	400	0	400
6	16384	200	0	200
7	32768	100	0	100
8	65536	10	0	10

```

value = 8 = 0x8

```

Figure 5. Memory query results

A test case was designed to test GDOS's memory management function. The test case applied for a 400-byte memory and a 600-byte memory respectively. According to the allocation rules of memory pool, 400-byte memory should occupy a memory block in the 512-byte memory pool, whereas 600-byte memory should occupy a memory block in the 1024-byte memory pool. After the test case is executed, memory information is showed in Fig. 5. It shows that the memory pool of 512 bytes and 1024 bytes is occupied by a block of memory respectively.

E. Test and Analysis of Layer 2 Protocol Module

Through communication testing in our 3G wireless network controller, a single DSP can support 70-route voice calls, with good and clear voice quality. After our 3G wireless network controller was tested for 72 hours with full load, GDOS ran stably without any malfunction. Fig. 6 shows the query information of layer 2 protocol module, where 'UciuNum' is the route number of voice calls, 'ScheTime' is the execution time for 20ms-task (ScheTime less than 20ms means normal, whereas greater than or equal to 20ms means abnormal.)

DepNo	Status	CellNum	UciuNum	ScheTime	MemRatio
1	MasterUnBlock	5	70	19	83
2	MasterUnBlock	0	0	0	0
3	MasterUnBlock	0	0	0	0
4	MasterUnBlock	0	0	0	0
5	MasterUnBlock	0	0	0	0
6	MasterUnBlock	0	0	0	0
7	MasterUnBlock	0	0	0	0
8	MasterUnBlock	0	0	0	0
9	MasterUnBlock	0	0	0	0
10	MasterUnBlock	0	0	0	0
11	MasterUnBlock	0	0	0	0
12	MasterUnBlock	0	0	0	0
13	MasterUnBlock	0	0	0	0
14	MasterUnBlock	0	0	0	0

value = 71 = 0x47 = 'G'

Figure 6. Query results of layer 2 protocol module

IV. APPLICATION AND EFFECT OF GDOS

GDOS has been successfully applied to a 3G wireless network controller developed by a large telecommunication company. It is widely used in Brazil, Romania, Libya and Hong Kong and other countries and regions as well as Beijing, Tianjin, Shenyang, Qingdao, Qinhuaangdao, Xiamen and other cities. The wireless communication system has been fully tested and affirmed by the users.

Applying GDOS to 3G radio network controller refines the DSP software development. GDOS developers are only responsible for the bottom development. GDOS improves the efficiency of software development, the legibility of the software architecture, and the maintainability of the system.

The strong portability of GDOS has been fully verified during its transplantation to TI C6482 and Intel Jasper Forest. A great deal of manpower was saved by using GDOS (ref. Table I & Table II. 'Traditional method' means that DSP system was developed directly based on the underlying platform).

TABLE I. TRANSPLANTATION COMPARISON FROM TI C6414 TO TI C6482

	Transplantation Codes (lines)	Transplantation Workload (man-month)
GDOS-based	3000	1
Traditional method	20000	7

TABLE II. TRANSPLANTATION COMPARISON FROM TI C6414 TO JASPER FOREST

	Transplantation Codes (lines)	Transplantation Workload (man-month)
GDOS-based	8000	3
Traditional method	60000	20

V. CONCLUSIONS AND DISCUSSIONS

DSP has been widely used since the emerging of high-performance low-cost DSP chips. But DSP software development starves for a unified operating system. A generic DSP-based real-time operating system (GDOS) is designed and implemented in this paper.

GDOS not only realized common functions of OS such as multi-task management, clock management, synchronization and mutex, but also implemented enhanced memory management function. Its log management and debugging function facilitates the maintainability and stabilizability of application systems. Furthermore, GDOS can reduce the development difficulty and improve the development efficiency of DSP application systems. GDOS plays an active role in the popularization of DSP applications.

Future work will consider implementing and customizing the functions of file system and peripheral management as well as the extending of supporting platform. We are trying to make GDOS be a stand-alone business system to serve more DSP applications.

REFERENCES

- [1] Andrew S. Tanenbraum, Albert S., Woodhull, Operating Systems Design and Implementation, Prentice Hall, Jan 2006.
- [2] B. Ackland, P.D Arcy, "A new generation of DSP architectures", Proceedings of the Custom Integrated Circuits Conference, 1999, pp.531-536.
- [3] Texas Instruments Inc., TI DSP BIOS User Manual and Driver Development, Qshinghua Publication Press, April 2007.
- [4] Dejan Minic, MSC8144 SmartDSP OS Ethernet Demonstration Using the Java GUI, Freescale Semiconductor Inc., May 2007.
- [5] P. Lapley, DSP Processor Fundamentals, IEEE Press, New York, 1997.
- [6] WANG Xi-lin, Design and implementation of the universal RTOS on DSP, East China Normal University, 2008.
- [7] SU Hong-qi, YANG Feng, "Research on high-accuracy programmable timer used in acquiring UWB signals", APPLIED GEOPHYSICS, 2005, 2: 127-130.
- [8] Klein, Netzer, Lu, "Detecting race conditions in parallel programs that use semaphores", Algorithmica, 2003, 35: 321-345.